VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Hybridní SOM pro překrývající se shlukování využívající rough set

Hybrid SOM for Overlapping Clustering with Rough Sets

2012

Bc. Martin Klapec

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Martin Klapec**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Hybridní SOM pro překrývající se shlukování využívající rough set
Hybrid SOM for Overlapping Clustering with Rough Sets

Zásady pro vypracování:

V rámci vědecko výzkumných článku se dnes můžeme dočíst o řadě implementací metod překrývajícího se shlukování. V rámci této diplomové práce student jednu z těchto metod bude implementovat. Vybraná metoda je založena na SOM a rough set.

Jednotlivé body práce jsou:
1. Prostudovat a provést experimentální implementaci SOM.
2. Prostudovat problematiku rough set.
3. Prostudovat metodu hybridních SOM pro překrývající se shlukování využívající rough set.
4. Implementovat metodu překrývajícího se shlukování.
5. Provést experimenty.
6. Zhodnocení výsledků experimentů.

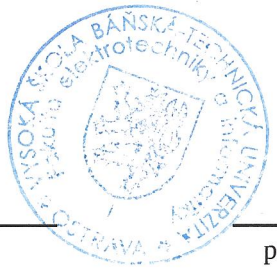
Seznam doporučené odborné literatury:

[1] Heaton J.: Introduction to Neural Networks for C#, ISBN-10: 1604390093, 2008
[2] Mohebi, E. (2008). Hybrid Self Organizing Map for Overlapping Clusters. Pattern Recognition, 1(1), 11 -20. Science & Engineering Research Support Center (SERSC)

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jan Martinovič, Ph.D.**

Datum zadání: 18.11.2011
Datum odevzdání: 04.05.2012



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 4. května 2012 .....

Na tomto místě bych rád poděkoval Ing. Janu Martinovičovi, Ph.D. za vedení diplomové práce, za jeho podporu a trpělivost. V neposlední řadě bych také rád poděkoval své rodině za podporu při studiu.

Abstrakt

Tato práce se zabývá hybridní samo organizující neuronovou sítí se shlukováním využívající rough sety. Nejprve popisuje jednotlivé části a poté jejich vzájemné spolupráce. V praktické části byla provedena implementace tohoto algoritmu a srovnána s běžným shlukovacím algoritmem K-means. Na závěr jsou shrnuty poznatky o těchto algoritmech nabyté při studiu této problematiky.

Klíčová slova: neuronové sítě, hybridní SOM, Kohonenovy mapy, Rough set, K-means, shlukování

Abstract

This work deals with hybrid self-organizing neural networks for clustering with rough sets. First of all, it describes the individual parts and then their mutual cooperation. In the practical part was the implementation of this algorithm made and compared with ordinary clustering K-means algorithm. The conclusion summarizes the findings of these algorithms gained during solving this problem.

Keywords: neural networks, hybrid SOM, Kohonen maps, Rough set, K-means, clustering

Seznam použitých zkratek a symbolů

RAM	–	Random Access Memory
SOM	–	Self Organizing Network
RGB	–	Red Green Blue
DoS	–	Denial of Service

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

V současnosti i v minulosti se lidstvo vždy snažilo uchovávat data, ať už zápisem událostí do knih a spisů, tak v nynější době populárního ukládání do velkých elektronických databází. Problém nastává, když je potřeba získat nějaké konkrétní informace o daném souboru. Například chceme pochopit vzájemné vazby a závislostí mezi daty. Jde o tvz. „vydolování“ užitečných informací z datových souborů. Jelikož objemy dat v databázích neustále narůstají, je tento úkol stále těžším a náročnějším na technické vybavení. Disciplína, která se zabývá tímto problémem se nazývá Data Mining. Jde o analytickou metodologii pro získávání informací z velkých datových souborů [2]. Obsahuje mnoho systémů a algoritmů, které pomáhají při tomto procesu.

Cílem této práce je ukázání a následné implementace jedné z metod pro získání užitečných informací z datových souborů na základě jejich vlastností a vzájemnou podobností datových prvků mezi sebou. Byla vybrána metoda, založena na shlukování dat s použitím samoorganizujících neuronových sítí a teorie rough set [12]. Byl proveden experiment, kde je tato metoda porovnána s běžnými metodami pro shlukování dat. Rozhodujícím faktorem je kvalita vytvořených shluků a schopnost správně klasifikovat datový soubor.

1.1 Struktura práce

Vlastní vypracování diplomové práce začíná kapitolou 2 a je rozvržena do několika následujících částí. V kapitole 2 jsou popsány neuronové sítě s pozvolným úvodem do problematiky. Je zde uvedena a „osvětlena“ zjevná souvislost s lidským mozkem, byly definovány základní pojmy a rozdělení těchto sítí. Detailní popis jedné neuronové sítě, která se použila v experimentu je v kapitole 3 nazvané Samoorganizující neuronová síť. Jsou zde popsány vlastnosti této sítě, princip funkce, metriky a matematické vzorce, které využívá. V kapitole 4 a 5 je rozebrán algoritmus K-means a rough set. Struktura těchto kapitol je velice podobná předešlé. Je zde uvedeno vše, co je důležité pro pochopení funkce. V kapitole 6 je názorně popsána spolupráce neuronové samoorganizující sítě, shlukovacího algoritmu K-means a rough set. V kapitole 7 jsou popsány metody, které byly použity pro vyhodnocení shlukování a popis datové kolekce použité v analýze. Kapitola 8 je vyhodnocení experimentu.

2 Neuronové sítě

V dnešní době počítačů, nových technologií a dramatického vývoje ve všech odvětvích informatiky a robotiky by se mohlo zdát, že počítače by mohly v blízké době dosáhnout úrovně lidského mozku a začít také samostatně uvažovat a rozhodovat, tak jak to umí zmíněný lidský mozek. Na první pohled, kdy i obyčejná kalkulačka [3] dokáže vypočítat druhou odmocninu ze dvou ve zlomku sekundy s přesností na devět desetinných míst. Proč by tedy nynější super počítač [4] nemohl nabýt vědomí a samostatně myslet? Trefně píše Jeff Heaton ve své knize [7] „faster is not always better for problem solving“ v překladu to znamená, že rychlejší není vždy lepší pro řešení nějakého problému. Pro srovnání si porovnejme dnešní normálně vybavený počítač (Intel i3 2,4GHZ, 4GB RAM) s mozkem 4letého dítěte. Je jasné, že dospělý člověk, natož dítě, nespočítá druhou odmocninu ze dvou tak rychle, a jestli vůbec, z hlavy, tak jako počítač. Ale když ukážeme dva obrázky dítěti a na jednom bude kočka a na druhém pes a zeptáme se ho na kterém je kočka. Dítě, pokud v životě už někdy tyto zvířata vidělo, okamžitě rozhodne. Pro počítač by byl tento úkol extrémně těžký [7]. Nicméně existují aplikace, které se inspirovaly lidským mozkem a které se běžně využívají. Říká se jim neuronové sítě.

2.1 Inspirace biologií

Lidský mozek vždy fascinoval vědce po celém světě. Byla a je stále touha poznat jeho funkci a z čeho se skládá. Neuronové sítě, které se používají pro různé úlohy našly inspiraci v lidském mozku. Snaží se ho vlastně jen simulovat, protože implementace skutečných neuronů a celé sítě je díky velké složitosti mozku extrémně náročná [7].

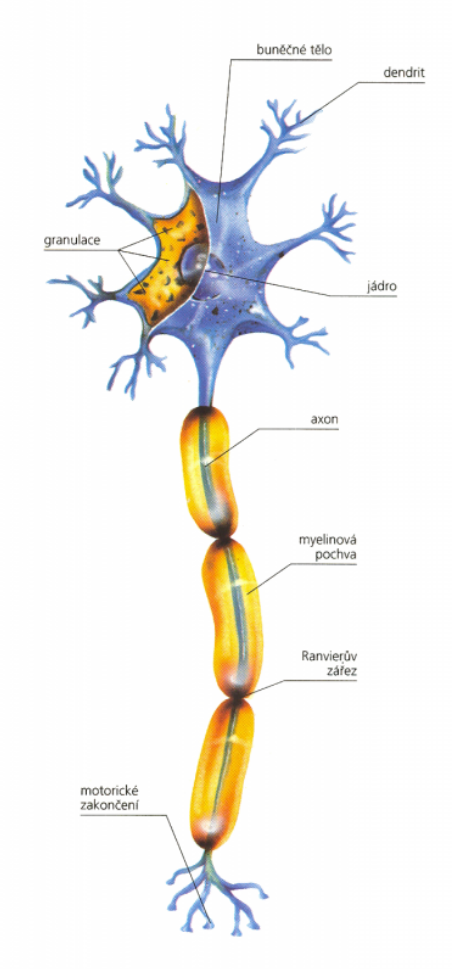
Základním kamenem mozku je nervová buňka nebo-li krátce neuron. Jak vypadá je znázorněno na obrázku 1.

Popis neuronu z biologického hlediska lze nejlépe pochopit z jeho fukce. Jako vstupy do neuronů slouží dendrity. Do nich se přivádí analogový signál z ostatních neuronů. Tyto signály se sečtou v jádře nebo-li Soma. Každý neuron má svou vnitřní aktivační funkci, což je vlastně nějaká limitní hodnota. Když součet vstupů přesáhne tuto hodnotu, neuron tzv. „vypálí“ nebo-li pošle signál skrz axon do dalších neuronů [7, 5]. Detailnější popis neuronu z biologického hlediska lze najít v této publikaci [10].

Na obrázku obrázku 2 je ukázán model neuronu pro použití v neuronových sítí, kde byla předlohou nervová buňka mozku.

Tento neuron má dva vstupy. Každý vstup má určitou váhu. Váha je číselná hodnota, která ohodnocuje vstup. Čím vyšší hodnota váhy, tím je větší pravděpodobnost překročení prahové hodnoty neuronu nebo-li aktivační funkce. V tomto příkladě má prahovou hodnotou s velikostí 2,5. I tento samostatný neuron můžeme nazvat neuronovou sítí s binárním rozhodováním (na vstup jdou pouze hodnoty 0 nebo 1 pro jednoduchost). Neuron zde simuluje chování logické funkce AND [6]. To znamená, že výstup bude nenulový, pokud na obou vstupech bude nenulová hodnota. Pravdivostní tabulka pro tuto funkci je uvedena v tabulce 1.

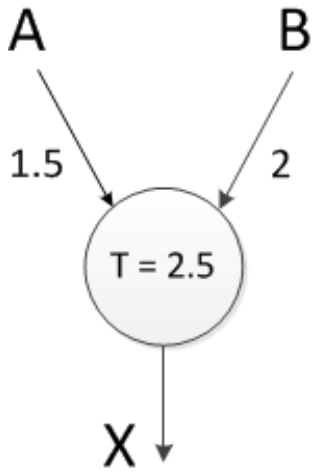
Postup při vyhodnocení výstupu pro případ, že A a B budou nulové by byl následující. Hodnoty každého vstupu se vynásobí s váhou daného vstupu a tyto vynásobené hodnoty



Obrázek 1: Nervová buňka [18]

se nakonec sečtou. Tedy pro A: 0 * 1,5 = 0 a pro B: 0 * 2 = 0 a součet vážených hodnot je 0 + 0 = 0. Výsledek se porovná s prahovou hodnotou neuronu. Lze jasně vidět, že jí zdaleka nepřekročil a že neuron neudělá nic. Tedy na výstupu bude stále nula. Pro případ, kdy na vstupech A a B budou jedničky se bude postupovat stejně. A: 1 * 1,5 = 1,5 ; B: 1 * 2 = 2 kde součet je 1,5 + 2 = 3,5. Zde hodnota překračuje prahovou hodnotu a neuron tzv. „vypálí“ a na výstupu bude jednička.

Pro jednoduché logické funkce, by jeden neuron stačil ale pro sofistikovanější využití se neurony spojují do větších sítí. Ani v mozku není jen jeden centrální neuron ale je jich mnohem více a to kolem 100miliard [5]. Jsou vzájemně propojeny mezi sebou. Při přirovnání s počítačem, kde procesor je hlavní výpočetní jednotka a pamětí je RAM paměť, která slouží k uložení hodnot si lze představit, že každý neuron je malá výpočetní jednotka (procesor) a spojení mezi neurony je pamětí RAM každého neuronu. Tato sít tvoří rozsáhlou paralelní strukturu. Paměti mezi neurony se říká váhy neuronu.



Obrázek 2: Model neuronu

A	B	X
0	0	0
0	1	0
1	0	0
1	1	0

Tabulka 1: Pravdivostní tabulka pro logickou funkci AND

Struktura neuronové sítě by nyní měla být trochu jasnější. Nyní je třeba říct, jak „naučit“ síť řešit nějaký problém. Obecně platí, že vhodné úkoly jsou ty, které nejdou vyjádřit jako jasná, definovaná série kroků. Tak jako se mozek učí novým věcem na základě nějaké minulé zkušenosti.

2.2 Rozdělení neuronových sítí

Základním rozdělením neuronových sítí, je podle jejich způsobu učení. Učení sítě je velmi důležitá část, protože ovlivňuje celou její následnou funkcionalitu.

2.2.1 Učení s učitelem

Toto učení neuronové sítě je specifické tím, že množina dat pro učení má ke vstupům odpovídající výstupy. Tato nesetříděná množina se rozdělí na dvě části. Na část trénovací a na část validační. Trénovací množina je určená pro naučení sítě nějaké funkci. Třeba predikci vývoje akcí na burze. Validační množina je pro ověření správnosti funkce dané neuronové sítě. Učení probíhá tak, že prvky množiny dat jsou předkládány neuronové síti i s odpovídajícími výstupy. Po každém předkladu se upraví vnitřní váhy neuronové sítě tak, aby při některém dalším předložení stejného (nebo velice podobného prvku) byla

schopna lépe rozpoznat jeho zařazení. Následující fáze je validace, kde na vstup sítě je předložena validační množina a výstupy sítě jsou porovnány s očekavanými výstupy. Více o problematice se je možné dočíst v [7].

2.2.2 Učení bez učitelem

Jde o samoorganizující neuronové sítě někdy nazývané kohonenovy mapy podle jeho tvůrce Teuvo Kohonena [1]. Typicky se tyto sítě používají pro shlukování nebo klasifikaci dat, kdy chceme vstupní množinu nějak rozdělit do skupin nebo najít závislosti mezi prvky dat.

Učení bez učitele je velice podobné jako učení s učitelem. Nyní ale nejsou dostupné očekávané výstupy k množině dat určené pro trénování sítě. Množina dat je opakovaně předkládána neuronové síti a síť si sama upravuje vnitřní váhy tak, aby při příštím předložením lépe klasifikovala stejný nebo podobný vzorek dat. Samoorganizující sítě zkráceně SOM jsou detailněji popsány v následující podkapitole číslo 3 Samoorganizující neuronové sítě.

Typické úkoly pro neuronové sítě jsou následující [7]:

Klasifikace
: Jde o rozdělení nějakého souboru dat do skupin. Například rozdělení žádostí o životní pojištění do různých rizikových skupin.

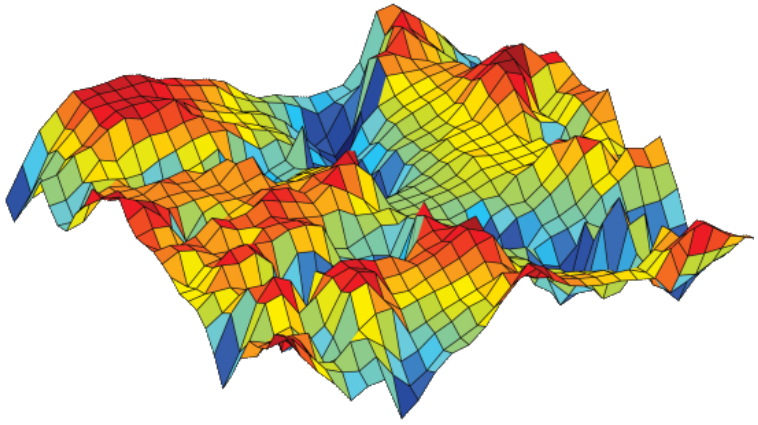
Predikce
: Odhadování následných kroků, hodnot apod. Podle zaznamenané historie dané veličiny. Ideálním příkladem je odhad ceny růstu či propadu akcií na burze v následujícím časovém období.

Pattern Recognition - Rozpoznávání vzoru
: Ideální úkol pro neuronové sítě. Jde například o rozpoznávání řeči, obrazu, vizuálních předloh, písma apod.

Tyto úlohy se dají řešit i pomocí jiných algoritmů a statistických metod. Neuronové sítě pro srovnání jsou v základu pomalejší v řešení ale jsou jednoduché na implementaci a díky výše zmíněnému paralelismus můžou výpočty probíhat na více než jednom procesoru a tím rapidně zvýšit rychlost.

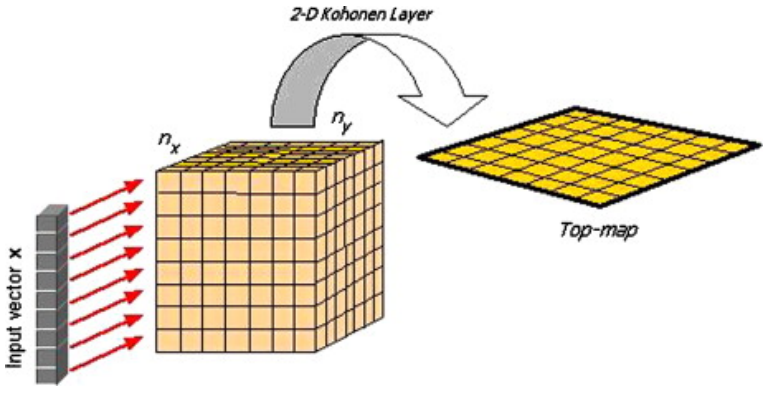
3 Samoorganizující neuronové sítě

Někdy nazývány Kohonenovy mapy podle tvůrce Teuvo Kohonena [1]. Jedná se o neuronovou síť, která patří do rozdělení sítí s učením bez učitele. To znamená, že učení nebo-li trénování sítě se neprovádí formou očekávaných výstupů jako u sítí s učitelem. Používá se tam, kde chceme zjistit nějaké závislosti mezi daty, které mají mnoho atributů. Z toho vyplývá, že tato neuronová síť dělá vlastně zobecnění těchto vysoko dimenzionálních dat do dvou nebo tří rozměrného prostoru, což lze jednoduše vizualizovat a vidět na obrázcích 3 a 4.



Obrázek 3: Příklad 3D vizualizace Kohonenovy mapy [19]

Například při klasifikaci žadatele o půjčku v nějaké bance do rizikových skupin musí žadatel uvést mnoho informací o sobě samém, o svém zaměstnání, finanční situaci, majetku apod. Vizualizace těchto dat přímo se všemi atributy by vedlo k vytváření X - dimenzionálního prostoru, který by byl velice nepřehledný a neměl žádnou informační hodnotu.

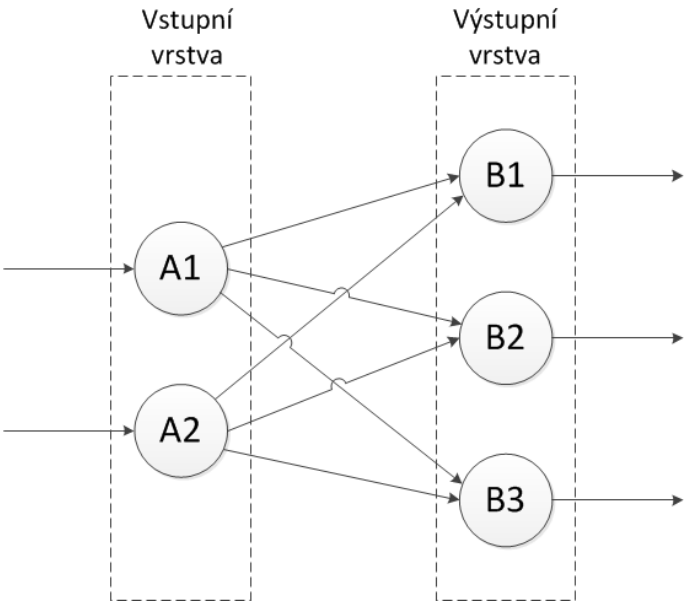


Obrázek 4: Jednoduché schéma zobecnění vstupních dat do samoorganizující sítě [16]

Na vstup sítě tedy přichází všechny relevantní údaje o žadateli. Síť toto vyhodnotí a určí, do které rizikové skupiny žadatel patří. Lze tedy odhadnout, že výstup sítě není poskládán z jednotlivých neuronů výstupní vrstvy ale vždy jen jeden neuron je prohlášen za tzv. vítěze. Každý výstupní neuron je tedy přiřazen k nějaké skupině, kterou reprezentuje.

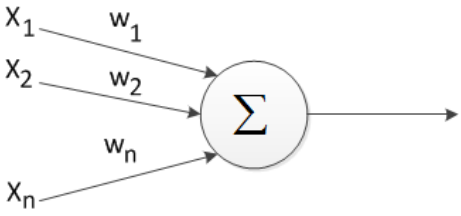
3.1 Struktura sítě

Jedná se o dvouvrstvou neuronovou síť, která je uvedena na obrázku 5. Má vstupní vrstvu a vrstvu výstupní. Nejsou zde žádné skryté vrstvy jak je tomu u sítí s učením s učitelem. Vrstvy jsou vzájemně propojeny tak, jak je uvedeno na obrázku 5. Každý neuron ve vstupní vrstvě je propojen s každým neuronem ve vrstvě výstupní ve směru Vstupní vrstva do Výstupní vrstva.



Obrázek 5: Struktura samoorganizující sítě

Typ neuronu používaný pro tento typ sítí funguje trochu jiným způsobem. Jeho model je uveden na obrázku 6.



Obrázek 6: Model neuronu výstupní vrstvy samoorganizující sítě

Jde vidět, že neuron ve výstupní vrstvě může mít jakýkoliv počet vstupů ale má jen jeden výstup. (na obrázku 5 je sice zobrazeno, že například neuron A1 má výstupy do dalších tří neuronů B1, B2, B3 ale hodnota výstupu je vždy stejná). Každý neuron ve výstupní vrstvě má u každého vstupu svou váhu. Z toho plyne, že do neuronů vchází vážené hodnoty vstupů. Neuron nemá žádnou vnitřní funkci, která by aktivovala neuron ale jen sčítá vážené hodnoty vstupů a posílá je dále na svůj výstup. Neurony ve vstupní vrstvě nemají váhy na svých vstupech. Slouží jako vstupy sítě, na který se posílá prvek datového souboru. Tedy atributy jednotlivých prvků. V příkladu žadatele o půjčku by to byly hodnoty jako věk, cílová částka apod.

3.2 Učení sítě

Jak bylo několikrát zmíněno, jedná se o síť s učením bez učitele. Neurony ve výstupní vrstvě jen sčítají vážené hodnoty na svých vstupech a posílají je dále na svůj výstup. Tedy hodnoty na vstupech se vynásobí s jejich váhy (například X1 * W1). Vstupní hodnota nelze měnit. Je jasné, že na vstup neuronové sítě dáváme data, které chceme klasifikovat, shlukovat či vizualizovat. V příkladu o žadateli o půjčku v bance není logické a hlavně správné měnit jeho parametry. S hodnotami, se kterými lze něco dělat jsou výše zmíněné váhy. Váhy si lze představit jako nositele informace nebo-li paměti. Pokud tedy budeme neuronové síti opakovaně předkládat stejného žadatele o půjčku resp. jeho atributy (ve fázi učení). Hodnota vah vítězného neuronu a neuronů v okolí se bude zvětšovat. Důvod je takový, že při příštím stejném nebo podobném žadateli, neuronová síť dokáže lépe určit do jaké rizikové skupiny žadatel patří. To to je základní princip učení samoorganizujících neuronových sítí. Váhy neuronů výstupní vrstvy se vlastně snaží podobat a přiblížit datům, která přicházejí na vstup sítě. Jedná se „namapování“ váhových hodnot na vstupy.

Před detailním vysvětlením procesu učení sítě, je nutné se seznámit s parametry a pojmy, které se v neuronových sítí používají, a které slouží k nastavení sítě.

Epochy
: Jedná se o celé číslo, které určuje počet cyklů vkládání celého datového souboru neuronové síti.

Parametr učení α
: Desetinné číslo, které ovlivňuje učení asi nejvíce. Pohybuje se většinou v rozmezí 0 až 1 ale některé publikace uvádějí -1 až 1 [7]. Jeho užití je hlavně ve výpočtech a úpravách vah neuronů výstupní vrstvy. Typicky je hodnota 0,4 až 0,5. Při vysoké hodnotě se síť učí velmi rychle ale může dojít k tvz. přetrénování, kde síť ztrácí schopnost zpracovat vstupní data. Při nízkých hodnotách se síť nemusí dostatečně naučit vzory předkládané síti a může je nepřesně zařadit.

Parametr okolí β
: Charakterizuje jak velké bude okolí vítězného neuronu. Neurony, které se nachází v tomto okolí budou mít upraveny váhy.

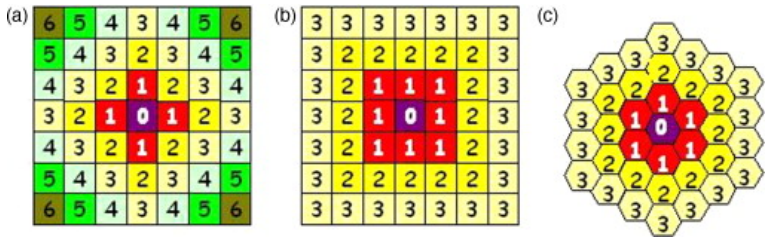
Počet výstupních neuronů
: Odpovídá počtu kategorií či skupin ,do kterých chceme data zařadit nebo shlukovat.

Počet vstupních neuronů

Počet neuronů ve vstupní vrstvě záleží na vstupním vektoru tzn. kolik atributů,sloupců mají jednotlivé prvky datového souboru.

Typ sítě

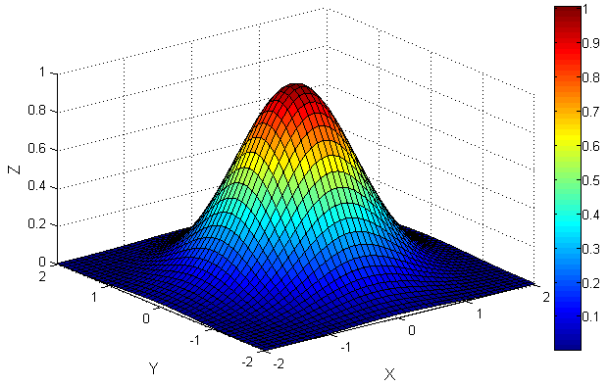
Jedná se vlastně o uspořádání neuronů ve výstupní vrstvě sítě. Příklady uspořádání jsou na obrázku 7.



Obrázek 7: Typ výstupní vrstvy (a)čtvercová, (b)obdelníková a (c)hexadecimální [16]

Způsob výběru okolí

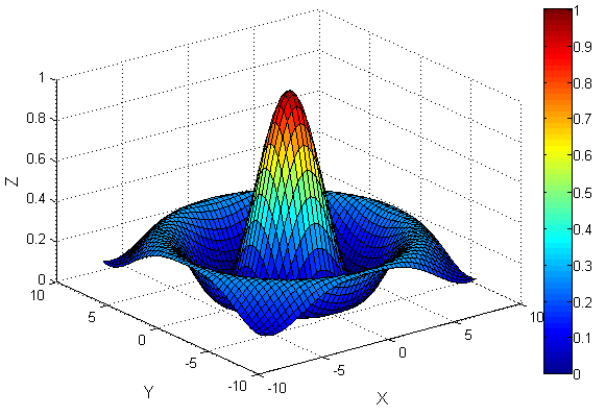
Souvisí s parametrem okolí. Jakmile je vybrán vítězný neuron, je třeba upravit váhy neuronů v jeho okolí. Váhy se upravují nejčastěji pomocí Gaussovy funkce. Její zobrazení je na obrázku 8.



Obrázek 8: Gaussova funkce [17]

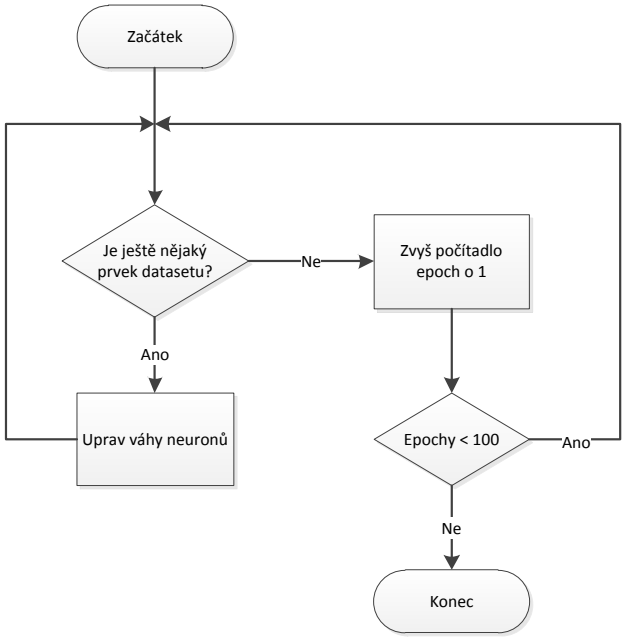
Na obrázku 8 je Gaussova funkce ve 2D. Můžeme si představit, že na vrcholu funkce je vítězný neuron. Jelikož je nejvýše, budou jeho váhy upraveny nejvíce. V okolí vítěze jsou další neurony, které nedosahují výšky vítěze, tedy jejich váhy budou upraveny méně.

Funkce Mexický klobouk uvedená na obrázku 9 je velice podobná gaussové funkci, kde na vrcholku „klobouku“ je vítězný neuron. Výška „klobouku“ ovlivňuje, jak moc budou váhy v okolí ovlivněny. Pro detailní popis a matematické vzorce je vhodné nahlédnout do [17].



Obrázek 9: Mexický klobouk [17]

Parametr učení a okolí se v průběhu učení sítě zmenšuje což umožní rychlé natrénování sítě v prvních epochách a poté jen pozvolné upravení vah neuronů. Nejde přesně říct, jaké parametry by se měly zvolit pro konkrétní síť či problém. Nicméně existují doporučené hodnoty nebo intervaly hodnot, které se nejčastěji používají. Více informací lze nalézt v publikacích [7, 8, 5].



Obrázek 10: Blokové schéma znázorňující proces učení. Upraveno z [7]

Na obrázku 10 je blokové schéma učícího procesu. Na začátku máme dataset, který se opakovaně předkládá neuronové síti. Po každém vložení se upraví váhy vítěze a váhy neuronů v jeho okolí. Po vložení posledního prvku datasetu, se počítadlo epoch zvětší o 1 a provede se kontrola, jestli počítadlo dosáhlo hodnoty předem definovaných epoch. Pokud ne, tak se celý dataset znovu předkládá neuronové síti. Učení končí při dosažení počtu epoch dané počítadlem.

Jde vlastně o soutěž výstupních neuronů. To znamená vyhraje ten neuron, který je podle euklidovské vzdálenosti nejblíže ke vstupnímu prvku. Váhy neuronů jsou před trénováním sítě nastaveny na náhodné malé hodnoty na intervalu 0 až 1. Při vložení vzorku na vstup sítě se počítá zmíněná euklidovská vzdálenost mezi vstupním prvkem a váhami jednotlivých neuronů podle vzorce 1.

$$\|d\| = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{1}$$

x_i váhy neuronu s indexem i
y_i váhy vstupu s indexem i

Ten neuron, který má nejmenší vzdálenost, je prohlášen za vítěze protože byl nejvíce podobný danému vstupu podle vzorce 2.

$$d_{min} = min\,\|d\| \tag{2}$$

Implementace v jazyce C# je uvedena ve výpisu 1.

```
protected double EuklidDistance(double[] input, double[] weights)
        {
            double euDistance = 0;
            double pom = 0;

            for (int i = 0; i < input.Length; i++)
            {
                pom = input[i] - weights[i];
                euDistance += pom * pom;
            }
            return Math.Sqrt(euDistance);
        }
```

Výpis 1: Implementace v jazyce C# : Euklidovská vzdálenost

Po nalezení vítěze se musí upravit jeho váhy a to podle vzorce 3.

$$w_{new} = w_{old} + \alpha\,[x - w_{old}] \tag{3}$$

w_{new} nové váhy neuronu
w_{old} staré váhy neuronu
α parametr učení
x vstupní vektor

Nyní se podle Gaussove funkce vypočte okolí neuronu podle vzorce a neurony, které se nacházejí v tomto okolí od vítězného neuronu se upraví váhy podle vzorce 4. Implementace je ve výpisu 2.

$$w_{new} = w_{old} + \alpha\beta\left[x - w_{old}\right] \tag{4}$$

w_{new} nové váhy neuronu
w_{old} staré váhy neuronu
α parametr učení
β parametr okolí
x vstupní vektor

Implementace v jazyce C# je uvedena ve výpise 2.

```
protected void AdjustWeights(Neuron winner, double[] input, double learningRate, double
    neighbourhood)
    {
        if (winner.Weights.Length == input.Length)
        {
            // Prochazeni vsech neuronu v siti
            for (int i = 0; i < rows; i++)
            {
                for (int j = 0; j < cols; j++)
                {
                    Neuron neuron = this.network[i, j];
                    double xs = (neuron.X - winner.X) * (neuron.X - winner.X);
                    double ys = (neuron.Y - winner.Y) * (neuron.Y - winner.Y);

                    //  vzdalenost mezi vitezem a aktualnim neurnonem (odmocnina se nedela
                        )
                    double distance = xs + ys;

                    // okoli se da na^2
                    double okoli2 = neighbourhood * neighbourhood;

                    if (distance < okoli2)
                    {
                        double[] neuronWeight = neuron.Weights;
                        double neighborhoodFact = NeighboorhoodFactor(winner, neuron,
                            neighbourhood);

                        //upraveni vah neuronu
                        for (int w = 0; w < neuronWeight.Length; w++)
                        {
                            double inputValue = input[w];
                            double oldWeight = neuronWeight[w];

                            double newWeight = oldWeight + learningRate * neighborhoodFact
                                * (inputValue - oldWeight);

                            neuronWeight[w] = newWeight;
```

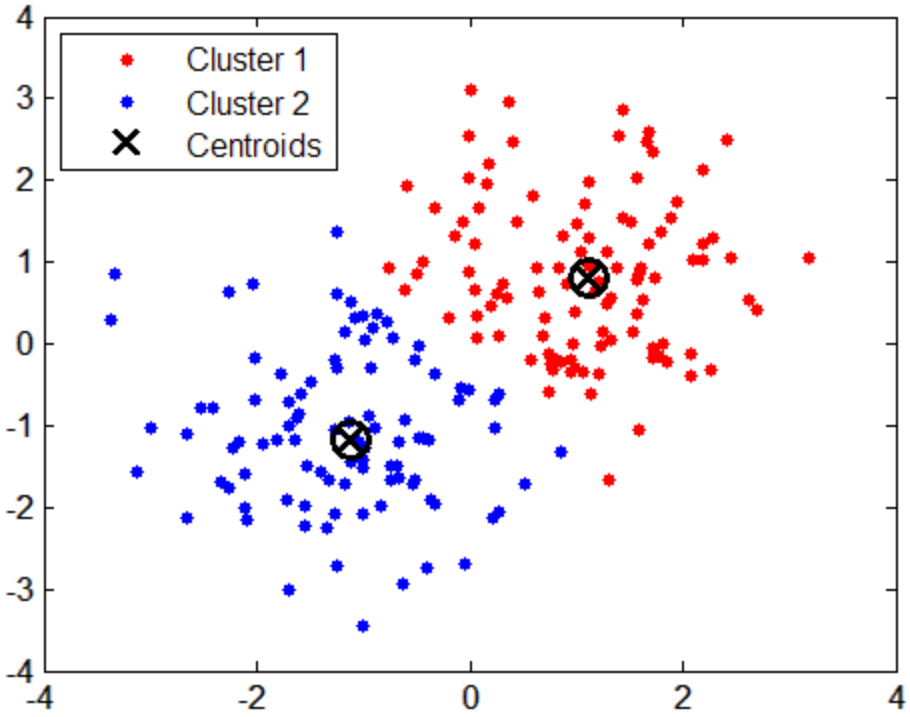
```
                        }
                        neuron.Weights = neuronWeight;
                    }
                }
            }
        }
        else
        {
            Console.WriteLine("winner.Weights.Length␣!=␣input.Length␣!!!!!");
        }
    }
```

Výpis 2: Implementace v jazyce C# : Opravení vah neuronů sítě

4 K-means

Patří mezi tvz. center-based shlukovací algoritmy [8]. Úkolem těchto algoritmů je rozdělení nějakého souboru dat do nepřekrývajících se skupin. Tedy každý prvek z datového souboru patří výhradně do jedné skupiny. Těmto skupinám se říká shluky. Data se dělí podle podobnosti jeden k druhému. Každý shluk je reprezentován svým středem. Tyto algoritmy jsou velice efektivní pro shlukování velkých vysoce dimenzionálních databází [8]. Níže je ukázán a popsán obyčejný K-means algoritmus, který byl v této práci použit.

Hlavním úkolem shlukování je redukce velikosti a složitosti datového souboru či nějaké databáze. Výskedkem je tedy usnadnění práce a zjednodušená manipulace se souborem. Redukce spočívá v nahrazení všech prvků shluku jedním, který nejvíce reprezentuje celý shluk [14].



Obrázek 11: Ukázka výsledku algoritmu K-means pro K = 2 [20]

K-means je jeden z nejpoužívanějších shlukovacích algoritmů. Byl navrhnut pro shlukování numerických dat, kde každý shluk má svůj střed. Počet těchto shluků je většinou definován uživatelem. Algoritmus má tedy za úkol přiřadit všechny objekty datového souboru do K shluků [13]. Nicméně existují metody, které se snaží řešit nalezení ideálního počtu shluků. Více o této problematice lze nalézt v publikaci [8].

```
1: function KMEANS(sada objektů N, počet shluků K)
2:          náhodná inicializace k objektů z N jako centra shluku
3:          repeat
4:                  for (pro všechny prvky v N)
5:                      jsou vypočteny euklidovské vzdálenosti mezi prvkem a jednotlivými centry
    shluků
6:                      prvek je přidán do shluku, k jehož centru má nejblíže
7:                  end for
8:              výpočet nových center shluků
9:          until (dokud nejsou nalezená stabilní centra shluků)
10: end function
```

Výpis 3: Pseudo algoritmus K-means

Na výpisu kódu číslo 3 je uveden Kmeans pseudo-algoritmus. Jako vstupy do algoritmu je dataset D, který se má shlukovat, počet K center, tedy počet žádaných shluků definovaných uživatelem a má dvě fáze:

Inicializační fáze

V této fázi jsou algoritmem náhodně vybrány prvky ze vstupního datasetu a dočasně prohlášeny za středy nebo-li centra shluků. Tento výběr proběhne jen jednou a to na začátku.

Iterační fáze

Zde se v cyklu postupně prochází celý dataset. Ke každému prvku datasetu se počítá euklidovská vzdálenost ke všem centrům, podle vzorce 5.

$$\|d\| = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{5}$$

Prvek datasetu je přiřazen do shluku ,k jehož centru má nejmenší vzdálenost podle vzorce 6.

$$d_{min} = min\,\|d\| \tag{6}$$

Po přiřazení všech prvků do svých shluků se provede přepočet hodnot všech center, podle vztahu 7.

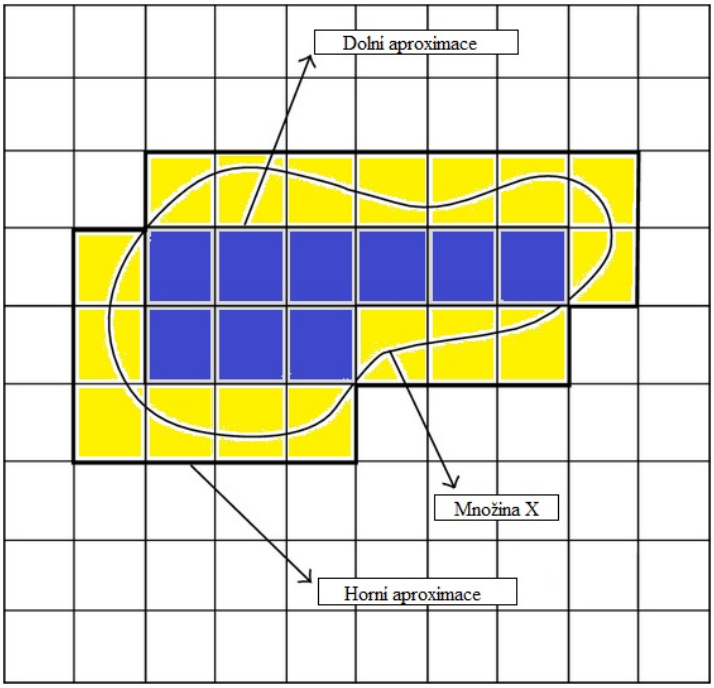
$$x_j = \frac{\sum_{v \in x} v_j}{m} \tag{7}$$

x_j shluk x s indexem j
v_j prvek datasetu s indexem j
m celkový počet prvků ve shluku j

Počítá se vlastně průměr všech parametrů prvků v příslušném shluku a tento průměr je vydělený četností prvků v shluku. V případě, že se nově přepočtené hodnoty center liší od hodnot předchozích, tak iterativní fáze probíhá od začátku s novými hodnotami center a dosavadní přiřazení hodnot datasetu je zrušeno. Algoritmus končí jen v případě, že hodnoty center zůstanou stabilní tzn. že se nezměnily.

5 Rough set

Jedná se o doplněk teorie množin vytvořený Z. Pawlakem v roce 1982 [15], který jej pojmenoval rough sety. Český název pro rough sety jsou hrubé množiny. Hlavní myšlenkou nebo spíše problém, který sloužil k inspiraci Z. Pawlaka bylo zjištění, že v mnoha aplikacích, kde je dána nějaká množinu objektů se nalézají nepřesné nebo neúplné údaje a my nejsme schopni doplnit nebo vyřešit tuto věc dostupnými metodami. Díky těmto vlastnostem některých aplikací dochází mezi daty k nerozlišitelnosti tzn. že neumíme přesně zařadit všechny prvky nebo objekty v nějakém souboru do svých skupin [15]. Dochází tedy k situaci, že prvek může patřit do více než jedné skupiny, protože není dostatek informací o daném prvku. Je proto výhodné stanovit hranice a zachytit, kdy s jistotou můžeme říct, že prvek patří jen do jedné skupiny a kdy patří i do více skupin. Na obrázku 12 jsou vyznačeny prvky množiny X, kde modré čtverečky patří jen do této skupiny a zatímco žluté čtverečky patří minimálně do jedné jinačí skupiny.



Obrázek 12: Reálná množina X a její zachycení pomocí Rough setu [21]

Teorie hrubých množin je považována za důležitou část umělé inteligence a kognitivních věd. Zvláště v oblastech strojového učení, v rozhodovacích procesech, získávání znalostí z databází, deduktivního chování a rozpoznávaní obrazu. Hlavní výhoda teorie rough setu je, že v datové analýze není potřeba provádět žádné statistické analýzy ani dopředu nic zpracovávat [15].

Dolní aproximace
Zde se nachází prvky souboru dat, které určitě patří do dané množiny tzn. že nemůžou být v žádné jiné aproximaci jakékoliv další skupiny.

Horní aproximace
Prvky datového souboru v této aproximaci můžou ale nemusí nutně patřit pouze k dané skupině. Můžou tedy patřit i do horních aproximací jiných skupin.

Hraniční oblast
Jde o rozdíl množin horní a dolní aproximace. O prvcích v této oblasti lze říci, že všechny také patří do minimálně jedné horní aproximace nějaké jiné skupiny.

Z textového popisu plynou následující podmínky pro určení zařazení prvku nějakého datového souboru do aproximací. Každá skupina má horní $\overline{A}(C_i)$ a dolní $\underline{A}(C_i)$ aproximaci, kde $\overline{A}(C_i)$ a $\underline{A}(C_i)$ pro každou skupinu $C \subseteq U$, kde U je soubor všech objektů platí:

$$\emptyset \subseteq \underline{A}(C_i) \subseteq \overline{A}(C_i) \subseteq U \tag{8}$$

$$\underline{A}(C_i) \cap \underline{A}(C_j) = \emptyset, i \neq j \tag{9}$$

Žádný prvek v dolní aproximaci skupiny C_i se nesmí objevit v žádné dolní aproximaci ostatních skupin.

$$\underline{A}(C_i) \cap \overline{A}(C_j) = \emptyset, i \neq j \tag{10}$$

Žádný prvek, který je v dolní aproximaci shluku C_i se nesmí objevit v žádné horní aproximaci ostatních skupin. Pokud nějaký prvek $u_k \subseteq U$ není v žádné dolní aproximaci, pak musí být v minimálně dvou horních aproximacích nějakých skupin.

6 Som a rough set

Dosud byly všechny technologie a použité struktury probrány zvlášť. V této kapitole bude popsána jejich vzájemná spolupráce a algoritmy, které byly použity.

6.1 Samoorganizující síť

Její hlavní funkcí je převedení vysoko dimenzionálních dat (tedy vektorů o hodně atributech) do prostoru s dvěma dimenzemi. Toto umožní snadnější práci s daty a menší náročnost na výpočetní výkon. Druhá funkce spočívá ve vytvoření matice, u které bude prováděno shlukování.

6.2 K-means a rough set

Nad maticí neuronové samoorganizující sítě bude probíhat shlukování pomocí K-means algoritmu se zakomponovanými rough sety. Algoritmu K-means byl popsán v kapitole 4 a jeho modifikace bude popsána dále. K-means algoritmus má za úkol rozdělit množinu prvků do počtu shluků definovaným uživatelem. Tedy jde přiřazování objektů do shluků podle jejich vzájemné podobnosti. Pro začlenění rough set teorie do tohoto algoritmus je nutné tento algoritmus pozměnit. Jak bylo popsáno v kapitole 5 shluk nebo-li skupina je rozdělena na dvě části, které se jmenují aproximace. První změna se týká přepočtu center shluků. Algoritmus K-means hledá své centra ve shlucích pomocí průměru všech prvků v daných shlucích. Změna je taková, že pro každý shluk se počítá průměr dolní aproximace a horní aproximace a tyto průměry jsou vynásobeny parametry w_{lower} a w_{upper}. Tyto parametry určují jak hodně se ovlivní změna nebo-li posun centra shluku. Vzorec pro výpočet centra je zde 11. Parametry w_{lower} a w_{upper} jsou čísla v intervalu 0 až 1. Většinou se volí v nerovnosti $w_{lower} > w_{upper}$ a jejich součet by měl být 1 [11].

$$x_j = \begin{cases} w_{lower} \times \frac{\sum_{v \in \underline{A}(x)} v_j}{|\underline{A}(x)|} + w_{upper} \times \frac{\sum_{v \in (\overline{A}(x) - \underline{A}(x))} v_j}{|\overline{A} - \underline{A}(x)|} & \text{if } \overline{A}(x) - \underline{A}(x) \neq \emptyset \\ w_{lower} \times \frac{\sum_{v \in) \underline{A}(x)} v_j}{|\underline{A}(x)|} \end{cases} \tag{11}$$

Lze vidět, že pokud hraniční oblast bude prázdná tedy $\overline{A}(x) - \underline{A}(x) \neq \emptyset$ pak sledovaný shluk bude běžným shlukem. Implementace v jazyce C# je ve výpisu 5.

```
public bool CalculationNewCentroids(List<NeuronRough> clusters) // kolekce shluků i se svými
    aproximacemi
    {
        int dimension = 0; // velikost vektoru
        int changeCentr = 0; // pocitadlo nalezenych
        foreach (NeuronRough leader in clusters)
        {
            dimension = leader.LeaderWeight.Length;
            double[] oldWeights = leader.LeaderWeight;//stare vahy leadra
            double[] newWeights = null; //nove vahy leadra
            List<NeuronRough> lowerApp = leader.LowerAppNeuronsArray; //kolekce neuronů
                v dolní aproximaci
            List<NeuronRough> upperApp = leader.UpperAppNeuronsArray; //kole

            if (lowerApp.Count != 0 && upperApp.Count == 0) //pokud je neco v dolni a neni
                nic v horni
            {
                newWeights = Helper.RoughtSum(lowerApp, w_lower, dimension); //soucet vah
                    * w_lower hodnota
            }
            else if (lowerApp.Count == 0 && upperApp.Count != 0)// pokud je neco v horni a
                nic v dolni
            {
                newWeights = Helper.RoughtSum(upperApp, w_upper, dimension); //soucet
                    vah * w_upper hodnota
            }
            else if (lowerApp.Count != 0 && upperApp.Count != 0) // pokud v obou
                aproximacich neco je
            {
                newWeights = Helper.SumArrays(Helper.RoughtSum(upperApp, w_upper,
                    dimension), Helper.RoughtSum(lowerApp, w_lower, dimension));

            }
            bool flagSAME = true;
            for (int i = 0; i < newWeights.Length; i++)
            {
                if (oldWeights[i] != newWeights[i])
                {
                    flagSAME = false;
                }
            }
            if (flagSAME) //pokud se vahy nezmenily
            {
                changeCentr++; //nalezen novy centroid
            }
            leader.LeaderWeight = newWeights;
        }
        if (changeCentr == clusters.Count)
            return false;
        return true;
    }
```

Výpis 4: Implementace v jazyce C# : Výpočet nových center shluků

Dalším modifikace K-means shlukovací algoritmu pro zakomponování rough setů je v rozhodování, do jaké aproximace bude prvek zařazen. Jestli do horní nebo do dolní aproximace aproximace shluku. Pro každý vstupní vektor nebo prvek v řekněme, že $d(v, x_i)$ je euklidovská vzdálenost mezi objektem a centru shluku X_i. Tedy pro každý objekt se vypočítá vzdálenost ke všem centrům shluků. $d(v, x_i) - d(v, x_j)$Rozdíl mezi těmito vzdálenostmi určuje vlastně příslušnost ke shlukům. Vybere se ta vzdálenost, která se je nejmenší. Tedy vzdálenost k nejbližšímu centru shluku. $d(v, x_i) = mind(v, x_j)$. Nyní následuje hlavní rozhodovací proces a to je ten, jestli $d(v, x_i) - d(v, x_j) \leq threshold\ i \neq j$. Pokud není nalezen rozdíl vzdáleností, který by tuto podmínku splňoval, pak testovaný prvek patří jenom do dolní aproximace shluku X_i. Pokud byl nalezen takový rozdíl vzdáleností, který této podmínce vyhovuje, pak je prvek přiřazen do horní aproximace shluku X_i a také do shluků, které podmínku splnily.

```
public void AssigningNeuronToBothAp(Neuron actualNeuron, List<NeuronRough> clusters)
        {
            double minDistance = Double.MaxValue;
            NeuronRough winLeader = null;
            foreach (NeuronRough leader in clusters)//1. nalezení nejbližší centra k danemu
                 neuronu
            {
                double distance = help.EuklidDistanceBetweenNeurons(actualNeuron, leader);
                if (distance < minDistance)
                {
                    minDistance = distance;
                    winLeader = leader;
                }
            }
            bool flagUpper = false;
            foreach (NeuronRough leader in clusters) //2. přiřazení neuronů shlukům
            {
                if (!leader.Equals(winLeader)) //aby nepřiřadil sebe samého
                {
                    double distance = help.EuklidDistanceBetweenNeurons(actualNeuron, leader);
                    double ratio = distance - minDistance;
                    if ( ratio <= threshold) //prirazeni do horni App actualniho leadra
                    {
                        leader.AddToUpperApp(new NeuronRough(actualNeuron, false));
                        flagUpper = true;
                    }
                }
            }
            if (flagUpper) //neuron byl přiřazen do horní approx. nějakého shluku
            {
                winLeader.AddToUpperApp(new NeuronRough(actualNeuron, false));
            }else //  pouze vítězný shluk má neuron ve své dolní approx.
            {
                winLeader.AddToLowerApp(new NeuronRough(actualNeuron, false));
        }   }
```

Výpis 5: Implementace v jazyce C# : Vkládání neuronů do dolních aproximací skupin

7 Experiment

V této částí práce jsou prováděny experimenty s neuronovými sítěmi, algoritmem K-means a rough sety. Budou podrobně zmíněny parametry všech algoritmů a neuronových sítí. Dále je zde vizuální ukázka algorimu K-means s rough sety, které jsou ohodnoceny validačními algoritmy. Bude i popsán soubor dat, který obsahuje dvě množiny pro testování a validaci. Na konci jsou výsledky srovnány a zhodnoceny.

7.1 Experiment s RGB

Zde bude názorně předveden algoritmus K-means s rozšířením o rough sety. Sloužil hlavně ke studiu problematiky rough setu a testování validačních indexů. Je ukázáno jak počet shluků a parametry rough setu ovlivňují vytvoření shluků.

Datovým souborem zde jsou náhodně vygenerované barvy v RGB kódu. Vstupním vektorem tedy jsou tři základní složky barvy. Důvod výběru kolekce barev je snadná vizualizace do dvourozměrného prostoru. Je natrénována pravidelná neuronová síť s čtvercovou topologií. Ve výstupní vrstvě sítě je 100 neuronů a ve vrstvě vstupní 3 neurony (RGB). Parametr učení je zvolen 0,5 a počet iterací 5000. Jako metrika je použita euklidovská vzdálenost a pro výběr okolí vítězného neuronu je použita gaussova funkce. Úkolem K-means algoritmu s rough sety je najít v natrénované síti shluky barev, které jsou si podobné a hlavně zachytit barvy, které by bez použití rough setu byly přiřazeny jen k jednomu shluku. Došlo by tak ke ztrátě informace. Parametry hrubých množin a počet zvolených shluků jsou v tabulce 4.

Počet K shluků	2	3	4	5
Threshold	0,1	0,2		
W_{lower}	0,65	0,9		
W_{upper}	0,35	0,1		
fuzier m	2			

Tabulka 2: Parametry rough setu pro názornou ukázku

7.1.1 Metody pro vyhodnocování shlukování

Jelikož shlukování je obecně proces, který není definovaný jasnou sérií kroků, není zde žádný univerzální postup jak rozhodnout, jestli shluky nalezené algoritmem jsou správné. Porovnání shlukovacích algoritmů nebo i nalezení optimálního počtu shluků je netriviální úkol [8]. Pro ohodnocení shlukování v tomto experimentu musí být požity metriky, které zohledňují případy, kdy prvky datového souboru můžou patřit i k jiným shlukům. Obecně tyto metriky lze rozdělit do dvou kategorií, kde první zohledňuje jen členství prvků ve shlucích a druhá zohledňuje jak členství, tak i datové prvky.

7.1.1.1 Matice příslušnosti Jedná se o matici prvků, která udává příslušnost k jednotlivým shlukům a je základem při počítání validačních indexů použitých v experimentu(viz. vzorec 12). V případě experimentu, kde shlukování bude probíhat u matice, kterou tvoří výstupní vrstva neuronů sítě, bude reprezentovat příslušnost těchto neuronů do shluků. Tato matice musela být upravena z důvodů použití hrubých množin. Byla upravena podle tabulky 3.

$$U = \sum_{i=1}^{n} \sum_{l=1}^{c} u_{li}^{m} \tag{12}$$

Hodnota příslušnosti	Případ
0	neuron nepatří do žádné aproximace shluku
0.5	neuron patří do horní aproximace shluku
1	neuron patří do dolní aproximace shluku

Tabulka 3: Modifikovaná příslušnost pro zakomponování do rough setu

7.1.1.2 Fukuyama-Sugeno Index Jedná se o index, který zohledňuje jak členství prvků ve shlucích, tak i jednotlivé datové prvky. Malé hodnoty tohoto indexu jsou znakem dobré kompaktnosti a dobré separace shluků. Více o indexu lze nalézt v pramenech [8].

$$V_{FS} = \sum_{i=1}^{n} \sum_{l=1}^{c} u_{li}^{m} (\|x_i - z_l\|^2 - \|z_l - z\|^2) \tag{13}$$

m fuzzy factor
u_{li}^{m} matice příslušnosti s exponentem m
z_l centrum shluku i
z centrum celého datasetu
$\|x_i - z_l\|^2$ vzdálenost mezi prvků ke svému shluku
$\|z_l - z\|^2$ vzdálenosti mezi centry shluků

```
public double FukuyamaSugenoIndex()
        {
            double INDEX = 0.0; //vysledek
            double uliNAm = 0.0;
            double yes = 1;      // patri do dolni aproximace
            double maybe = 0.5; // patri do horni aproximace
            int clusterNumber = −1;
            foreach (NeuronRough cluster in leaders)
            {
                clusterNumber++;
                double[] clusterMEAN = clustersMEANS.ElementAt(clusterNumber); // data je
                    prumer jednotlivych klastru
                double Zminuses = SquaredEuklid(clusterMEAN, datasetMEAN); ; // je to
                    euklidovska vzdalenost mezi prumerem klastru a prumerem datasetu na
                    druhou
```

```
            uliNAm = Math.Pow(yes, fuzier);       // Uli^m pro dolni app

            foreach (NeuronRough neuron in cluster.LowerAppNeuronsArray)
            {
                double inThebracket = (SquaredEuklid(neuron.Neuron.Weights, clusterMEAN)
                    - Zminuses);
                INDEX += uliNAm * inThebracket;
            }
            uliNAm = Math.Pow(maybe, fuzier);     // Uli^m pro horni app
            foreach (NeuronRough neuron in cluster.UpperAppNeuronsArray)
            {
                double inThebracket = (SquaredEuklid(neuron.Neuron.Weights, clusterMEAN)
                    - Zminuses);
                INDEX += uliNAm * inThebracket;
            }
        }
        return INDEX;
    }
```

Výpis 6: Implementace v jazyce C# : Výpočet Fukuyama-Sugeno indexu

7.1.1.3 Xie-Beni Index Jedná se o index, který měří kompaktnost a oddělenost utvořených shluků z datasetu. Kompaktnost je podíl celkové rozptýlenosti prvků ve shlucích k celkovému počtu prvků datasetu . Malá hodnota tohoto indexu vypovídá o správném shlukování. Je ale nutné říci, že hodnota indexu má klesající tendenci při neúměrném zvyšování počtu shluků. [8].

$$S = \frac{\sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^2 \left\| V_i - x_j \right\|^2}{n \min_{1 \le i \le j \le c} \left\| V_i - V_j \right\|^2} \tag{14}$$

```
private double Separation()
    {
        double minimalDistance = Double.MaxValue;
        NeuronRough pomClus = null;
        bool  firstIter  = true;

        foreach (NeuronRough cluster in leaders) //hledani nejmensi vzdalenosti center
        {
            if  (  firstIter  )
            {
                pomClus = cluster;  // inicializace  prvniho shluku
                 firstIter  = false;
                continue;
            }
            double pomDistance = euklidNA2(pomClus.LeaderWeight, cluster.LeaderWeight);

            if  (pomDistance < minimalDistance)
            {
```

```
                minimalDistance = pomDistance;
            }
        }
        return minimalDistance;
    }
```

Výpis 7: Implementace v jazyce C# : Výpočet separace pro Xie-Ben index

```
public double XieBeniIndex()
    {
        double sigma = 0.0; // sigma
        double uliNA2 = 0.0;
        double[] leaderWeigh = null;
        double yes = 1;      // patri do dolni aproximace
        double maybe = 0.5; // patri do horni aproximace
        int allneuronsCount = 0;
        int clusterNumber = -1;

        foreach (NeuronRough cluster in leaders)
        {
            clusterNumber++;
            leaderWeigh = cluster.LeaderWeight; // vahy leadra nebo-centra shluku
            allneuronsCount += cluster.LowerAppNeuronsArray.Count + cluster.
                UpperAppNeuronsArray.Count; //pocet vsech neuronu v klastru se
            // pricte do celkoveho poctu pridani do celkove
            uliNA2 = yes * yes;      // Uli^2 pro dolni app

            foreach (NeuronRough neuron in cluster.LowerAppNeuronsArray)
            {
                sigma += (uliNA2 * euklidNA2(leaderWeigh, neuron.Neuron.Weights));
            }
            uliNA2 = maybe*maybe;   // Uli^2 pro horni app
            foreach (NeuronRough neuron in cluster.UpperAppNeuronsArray)
            {
                sigma += (uliNA2 * euklidNA2(leaderWeigh, neuron.Neuron.Weights));
            }
        }

        double Fi = sigma / allneuronsCount; // tedka mame Fi
        double S = Separation();
        return (Fi / S);
    }
```

Výpis 8: Implementace v jazyce C# : Výpočet Xie-Beni indexu

Pomocí hodnot v tabulce 4 byly vytvořeny testovací sady. Po ukončení každé testovací sady byly zaznamenány indexy Xie-Beni a Fukuyama-Sugeno vloženy do tabulek 4 a 5.

	Threshold = 0,1		Threshold = 0,2	
k	Fukuyama-Sugeno	Xie-Beni	Fukuyama-Sugeno	Xie-Beni
2	-55,234	0,567	-49,110	0,337
3	**-56,2074**	0,232	-51,655	0,159
4	-56,041	**0,169**	-48,761	0,233
5	-51,388	0,191	**-54,340**	**0,151**

Tabulka 4: Výsledné Xie-Beni a Fukuyama-Sugeno indexy pro shlukování barev s hodnotami $W_{lower} = 0,65$ $W_{upper} = 0,35$

	Threshold = 0,1		Threshold = 0,2	
k	Fukuyama-Sugeno	Xie-Beni	Fukuyama-Sugeno	Xie-Beni
2	-54,609	0,323	**-51,991**	0,193
3	-54,819	0,148	-51,513	**0,107**
4	-57,346	0,174	-50,099	0,152
5	**-57,844**	**0,135**	-49,005	0,119

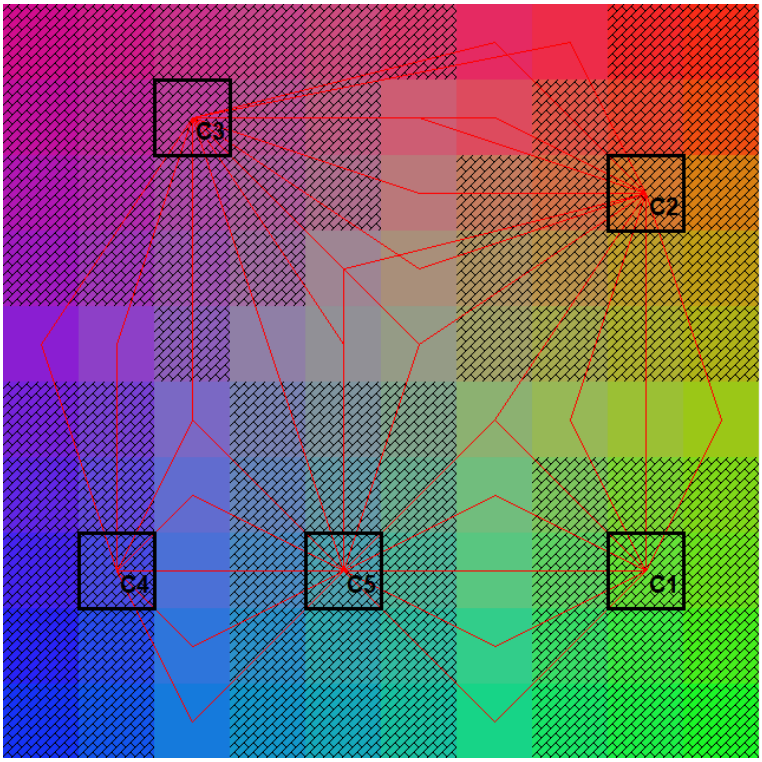
Tabulka 5: Výsledné Xie-Beni a Fukuyama-Sugeno indexy pro shlukování barev s hodnotami $W_{lower} = 0,9$ $W_{upper} = 0,1$

Tučně zvýrazněné hodnoty indexů v tabulkách 4 a 5 jsou nejlepší dosažené hodnoty v testovaných scénářích. Lze vidět, že pomocí těchto hodnot nebylo možné přesně určit optimální počet shluků, jelikož výsledky se liší. Nicméně u většiny je rozdíl řádově v desetinách jednotek. Vhodný počet shluků by se měl také určit podle požadavků zadavatele nebo typu úkolu. Například pro počet 5 shluků a hodnotami threshold = 0,1 a $W_{lower} = 0,9$ $W_{upper} = 0,1$ byl nalezen velice úzká hraniční část mezi shluky. Vizualizace této sítě je na na obrázku 13. Čtverce s číslem značí shluky. Mřížkovaná oblast kolem jednotlivých shluků značí neurony, která se nacházejí v dolní aproximaci shluků. Červené orientované linky značí neurony, které jsou v horní aproximacích shluků.

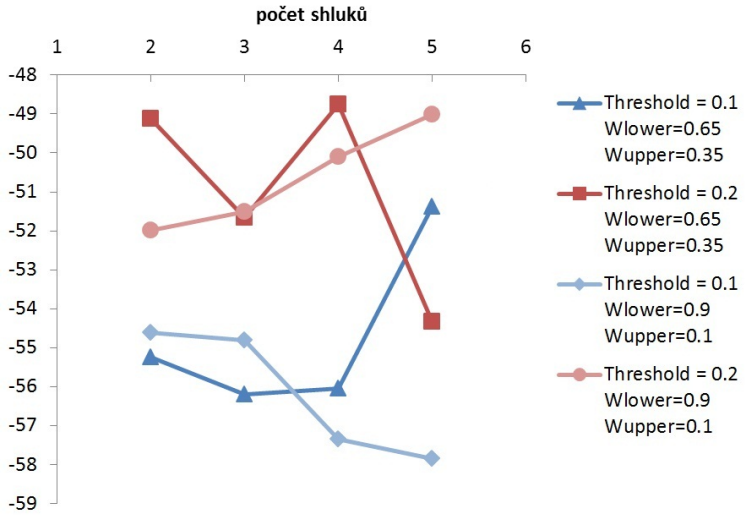
Na grafu 15 lze vidět, že index Xie-Beni má klesající tendenci. Toto značí, že shluky jsou dobře vzájemně odděleny. U grafu 14 Fukuyama-Sugeno indexu, který značí kompaktnost a oddělenost shluků lze pozorovat kolísání jeho hodnot pro různé parametry.

Na obrázku 16 jsou zobrazeny dvě identické sítě nad kterými bylo provedeno shlukování se stejnými parametry až na odlišnou hodnotu thresholdů. Lze jasně vidět, že tento parametr ovlivňuje nejvíce hranice shluků. Při hodnotě 0,1 (levá síť) se algoritmus více „snažil“ přiřazovat neurony do dolních aproximací shluků a neurony, které byly na hranici mezi dvěma shluky, byly přiřazeny do horních aproximací příslušných shluků. Při hodnotě 0,2 lze pozorovat významný rozdíl mezi v příslušnosti neuronů do aproximací shluků. Algoritmus zařadil více neuronů do horních aproximací shluků.

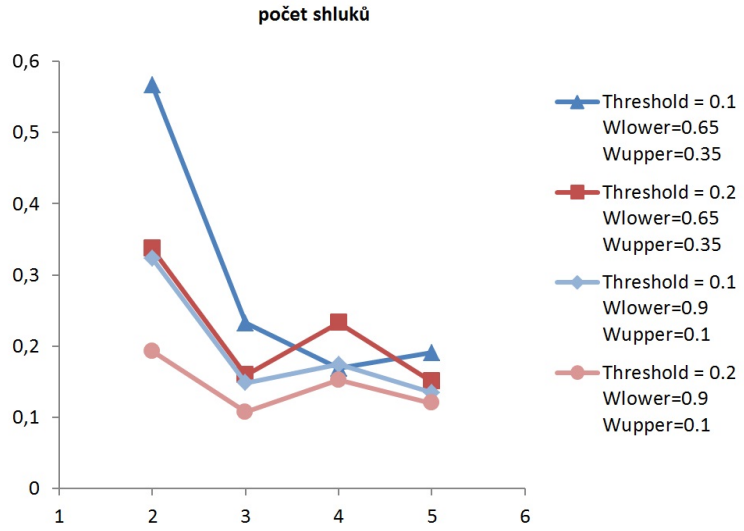
Tato ukázka sloužila pro řádné nastudování problematiky rough setu s algoritmem K-means. Bylo důležité vyzkoušet spolupráci tohoto algoritmu s neuronovou sítí a ověřit nabyté znalosti. Původně byl tenhle příklad zamýšlen jako hlavní experiment, který by se dal jednoduše vizualizovat. Nicméně byla zvolena množina dat, která je reálným



Obrázek 13: Vizualizace ukázkového příkladu s 5 shluky, threshold = 0,1 a hodnotami $W_{lower} = 0,9$ $W_{upper} = 0,1$



Obrázek 14: Graf Fukuyama-Sugeno indexu pro ukázkový příklad



Obrázek 15: Graf Xie-Beni indexu pro ukázkový příklad

problémem a je blíže praktickému využití. Indexy používané v tomto příkladě se nebudou dále používat. Ukázalo se, že index Xie-Beni má nastavovací parametr zvaný fuzzier, který by vyžadoval dodatečné netriviální experimentování a testování funkčnosti indexu.

7.2 Neuronová síť

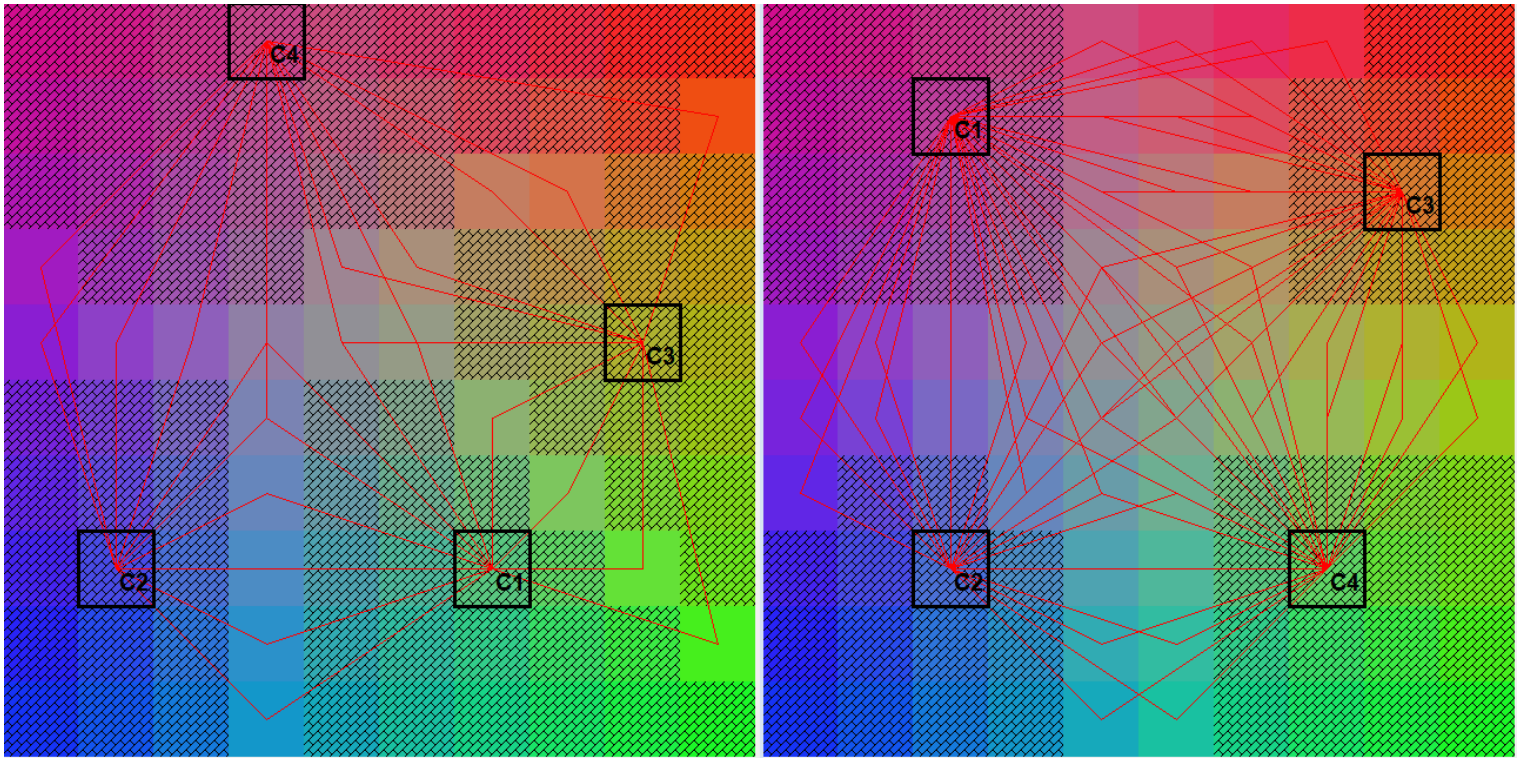
Bude popsána datová kolekce, definovány parametry, podle kterých se připraví neuronové sítě, které jsou použité v následujících experimentech. Dále je zde i popsáno jakým způsobem jsou sítě ohodnoceny.

7.2.1 Popis datové kolekce

V experimentu byla použita datová kolekce z balíku kolekcí Intruders [9]. Datová kolekce obsahuje kolekce dat jedné třídy síťového útoku. Obsahuje dvojici souborů. Soubor určený pro učení sítě, který obsahuje 5092 vektorů o 42 atributech a soubor určený k validaci a testování sítě obsahuje 6890 vektorů také o 42 atributech. Jednotlivé vektory popisují síťový provoz v rozsahu 0-1. Není tedy nutné datovou kolekci normalizovat. Při testování bylo použito jen 41 atributů. Doplňkový atribut, který se nachází na první pozici každé vektoru, slouží k určení, jestli se jednalo o útok (hodnota 1) nebo o běžný provoz (hodnota 0) [9].

7.2.2 Parametry neuronové sítě

Pro potřeby experimentu jsou vytvořeny neuronové sítě s různými počty neuronů a nastavovacích parametrů. Jejich kombinace je uvedena v tabulce 6.



Obrázek 16: Srovnání shlukování při threshold=0,1 a 0,2 se 4 shluky, $W_{lower} = 0,9$ $W_{upper} = 0,1$

Počet neuronů	25	49	100	196
Počet Epoch	200	500	1000	
Parametr učení	0.2	0.5		

Tabulka 6: Parametry k trénování sítě

Tvar neuronové sítě bude čtvercový. Parametr okolí je určen podle počtu neuronů ve výstupní vrstvě sítě tzn. podle druhé odmocniny z tohoto počtu. Jako metrika určování vzdáleností mezi neurony je použita euklidovská vzdálenost. Pro výběr okolí kolem vítězného neuronu je použita gaussova funkce. Popis parametrů je v kapitole 3.2.

7.2.3 Validace neuronové sítě

Učení sítě je popsáno v kapitole 3. Po naučení sítí jsou tyto sítě otestovány validační množinou Intruders. Měří se úspěšnost klasifikace prvků z této množiny a vlastně určení, zda se jedná útok nebo o normální běžný provoz. Toto ohodnocení je rozděleno do dvou fází.

1.Fáze

Úkolem této fáze je zjištění jaké hodnoty, budou výstupní neurony sítě reprezentovat. V tomto případě, jestli konkrétní neurony budou značit útoky nebo běžný provoz. Na vstup sítě se posílají prvky z validační množiny. Síť pokaždé určí jeden neuron, který se nejvíce podobá prvku na vstupu sítě. Každý neuron obsahuje počítadlo, které má před validací hodnotu nula. Pokud na vstupu je prvek, který značí útok, neuron, který „vyhraje“ zvýší své počítadlo o jedničku. Pokud vstupní prvek je normálním síťovým provozem, tak neuron zmenší hodnotu počítadla o jedničku. Po klasifikaci celé validační množiny se rozhodne, které neurony budou reprezentovat útok a které běžný síťový provoz. Toto rozhodnutí je dáno hodnotou počítadel jednotlivých neuronů. Pokud mají kladné hodnoty, neurony jsou reprezentanty útoku. Při záporných hodnotách se bude jednat o běžný provoz.

2.Fáze

V této fázi síť opět klasifikuje validační množinu. Nyní si je ale vědoma, které neurony reprezentují útok a které běžný provoz. Měřené hodnoty i s jejich zkratkou, která je použita v tabulkách jsou uvedeny v tabulce 7.

GA	Počet správě klasifikovaných útoků
GN	Počet správě klasifikovaného běžného provozu
BA	Počet špatně klasifikovaných útoků
BN	Počet špatně klasifikovaného běžného provozu
TG	Celkový počet správně klasifikovaných vstupů
TG[%]	Celkový počet správně klasifikovaných vstupů v %
TB	Celkový počet špatně klasifikovaných vstupů
TB[%]	Celkový počet špatně klasifikovaných vstupů v %

Tabulka 7: Měřené hodnoty a jejich zkratky

Parametr učení	Epochy	GA	GN	BA	BN	TG	TG[%]	TB	TB[%]
0.2	200	3256	1804	946	884	5060	73,44	1830	26,56
	500	3054	2373	1148	315	5427	78,77	1463	21,23
	1000	3327	2241	875	447	5568	**80,81**	1322	19,19
0.5	200	3918	1679	284	1009	5597	**81,23**	1293	18,77
	500	4169	1172	33	1516	5341	77,52	1549	22,48
	1000	4129	1292	73	1396	5421	78,68	1469	21,32

Tabulka 8: Naměřené hodnoty klasifikace pro síť o 25 neuronech

Parametr učení	Epochy	GA	GN	BA	BN	TG	TG[%]	TB	TB[%]
0.2	200	4127	1596	75	1092	5723	**83,06**	1167	16,94
	500	2824	2504	1378	184	5328	77,33	1562	22,67
	1000	4202	428	0	2260	4630	67,20	2260	32,80
0.5	200	4195	779	7	1909	4974	72,19	1916	27,81
	500	3663	2164	539	524	5827	**84,57**	1063	15,43
	1000	3429	2179	773	509	5608	81,39	1282	18,61

Tabulka 9: Naměřené hodnoty klasifikace pro síť o 49 neuronech

Parametr učení	Epochy	GA	GN	BA	BN	TG	TG[%]	TB	TB[%]
0.2	200	2917	2638	1285	50	5555	80,62	1335	19,38
	500	3853	1728	349	960	5581	81,00	1309	19,00
	1000	3456	2387	746	301	5843	**84,80**	1047	15,20
0.5	200	2714	2333	1488	355	5047	73,25	1843	26,75
	500	3749	2343	453	345	6092	**88,41**	798	11,58
	1000	3511	2396	691	292	5907	85,73	983	14,27

Tabulka 10: Naměřené hodnoty klasifikace pro síť o 100 neuronech

Parametr učení	Epochy	GA	GN	BA	BN	TG	TG[%]	TB	TB[%]
0.2	200	3271	1956	931	732	5227	75,86	1663	24,14
	500	3916	2092	286	596	6008	**87,20**	882	12,80
	1000	3764	2038	438	650	5802	84,21	1088	15,80
0.5	200	3269	2365	933	323	5634	**81,77**	1256	18,23
	500	3418	2071	784	617	5489	79,67	1401	20,33
	1000	3776	1667	426	1021	5443	79,00	1447	21,00

Tabulka 11: Naměřené hodnoty klasifikace pro síť o 196 neuronech

O naměřených hodnotách uvedených v tabulkách 8,9,10 a 11 můžeme říci, že nejlepší rozpoznávací schopnost má síť o 100 neuronech, která byla učena v 500 iteracích a měla učící parametr 0,5. Síť dokázala rozpoznat a správně zařadit 6092 prvků validační množiny o celkovém počtu 6890. Tedy měla 88,41 % úspěšnost. Můžeme také vidět, že všechny hodnoty pro síť o 100 neuronech vykazují úspěšnost (až na jednu vyjímku) více než 80%. Nejhorší síť v měření byla síť o 49 neuronech, která byla učena ve 1000 iteracích s parametrem učení 0,5. Navzdory nízké celkové úspěšnosti, jako jediná síť určí všechny útoky bez chyby. Nicméně v klasifikaci běžného provozu má největší chybovost.

7.3 K-means a neuronová síť

V tomto experimentu bude měřena klasifikační schopnost neuronové sítě v kombinaci algoritmu K-means na množině Intruders. Sítě, nad kterými bude prováděno shlukování jsou stejné jako u předešlého experimentu s parametry uvedené v tabulce 6. Je zkoumáno, jestli v síť v kombinaci s algoritmem K-means, lze dosáhnout lepších výsledků klasifikace množiny Intruders. Tento experiment je rozdělen na dvě menší části podle počtu neuronů sítí. V první části se testují sítě o 25 a 49 neuronech a v druhé sítě o 100 a 196 neuronech. Důvodem pro toto rozdělení je možnost použít vyšší počet shluků pro větší sítě. Počty shluků, které jsou použity jsou uvedeny v tabulce 12.

Počet neuronů	Počet center
25	3,6,9,12
49	3,6,9,12
100	5,10,15,20,25,30
196	5,10,15,20,25,30

Tabulka 12: Počet K shluků pro různé sítě

7.3.1 Validace kombinace neuronových sítí a K-means

Neuronové sítě jsou připraveny z předešlého experimentu v kapitole 7.2.2. Shlukování pomocí algoritmu K-means je prováděno nad výstupní vrstvou těchto neuronových sítí. Opět je nutné nejprve zjistit, jestli neurony reprezentují útok nebo běžný provoz. Je také použito ohodnocení z předešlého experimentu, které je popsáno v kapitole 7.2.3. Reprezentace neuronů jsou tedy již známé. Výstupní vrstva neuronová sítě tvoří matici, nad kterou se provede shlukování. Algoritmus rozdělí tuto matici do shluků. Vzniknou tak skupiny podobných neuronů. Rozhodnutí, co budou jednotlivé shluky reprezentovat závisí na typu neuronů ve shlucích. Pokud ve shluku je více neuronů, které reprezentují útok, tak celý shluk (a tím i všechny neurony) budou označeny za reprezentanty útoky. Při vlastním měření se posílají prvky validační množiny Intruders na vstup sítě. Porovnání, jestli se jedná o správnou klasifikaci vstupu se provádí mezi tímto vstupem (hodnotou, kterou reprezentuje) a shlukem ve kterém se výherní neuron nachází. Měřené hodnoty byly zaznamenány v tabulkách 16,17,18 a 19 , které jsou pro svou velikost umístěny v příloze. Měřené hodnoty i s jejich zkratkami použité v tabulkách jsou uvedeny v tabulce 15. Pro velkou velikost byly tabulky s naměřenými hodnotami a zkratkami přesunuty do přílohy.

7.3.2 Hodnocení

O naměřených hodnotách v tabulkách 16,17,18 a 19 lze říci, že pokud srovnáme nejlepší výsledky podle velikosti sítě zjistíme, že úspěšnost klasifikace kombinace K-means algoritmu a shlukovacích sítí je mírně menší. Například algoritmus K - means (K = 12) u sítě o

25 neuronech a dokázal úspěšně zařadit a rozpoznat 3058 útoků a 2366 běžného provozu z celkového počtu 6980 prvků což je podíl 78,72 %. Při porovnání s nejlepší neuronovou sítí vychází algoritmus mírně hůře a to o 2,09 %. Nejlepší hodnoty v tomto experimentu dosáhl algoritmus K-means (K = 25) u sítě o 100 neuronech. Jeho úspěšnost byla 88,41 %.

7.4 Rough set a neuronová síť

V tomto experimentu bude testována schopnost klasifikovat datový soubor Intruders pomocí kombinace neuronových sítí a algoritmu K-means se zakomponovanými rough sety. Lépe řečeno, úkolem je identifikace problémových záznamů, u kterých nelze jasně určit příslušnost. V tomto případě tedy rozhodnutí zda se jedná o útok nebo o běžný provoz. Neuronové sítě jsou stejné jako u experimentu K-means a neuronové sítě v podkapitole 7.3. Experiment byl rozdělen do dvou částí podle velikosti sítě podle tabulky 13 a 14. Důvodem je možnost zadání vyššího počtu shluků u větších sítí.

Počet K shluků	2	3	4
Threshold	0,1	0,2	
W_{lower}	0,65	0,9	
W_{upper}	0,35	0,1	

Tabulka 13: Parametry K-means algoritmu a rough setu pro sítě o 25 a 49 neuronů

Počet K shluků	6	8	12
Threshold	0,1	0,2	
W_{lower}	0,65	0,9	
W_{upper}	0,35	0,1	

Tabulka 14: Parametry K-means algoritmu a rough setu pro sítě o 100 a 196 neuronů

7.4.1 Validace kombinace neuronových sítí a K-means s Rough sety

Neuronové sítě jsou opět připraveny z předešlého experimentu v kapitole 7.2.2. Shlukování pomocí algoritmu K-means s rough sety je prováděno nad výstupní vrstvou těchto neuronových sítí. Opět je nutné nejprve zjistit, jestli neurony reprezentují útok nebo běžný provoz. Je také použito ohodnocení z předešlého experimentu, které je popsáno v kapitole 7.2.3. Reprezentace neuronů jsou tedy již známé. Výstupní vrstva neuronová sítě tvoří matici, nad kterou se provede shlukování. Algoritmus rozdělí tuto matici do shluků. Rozdělení je popsáno v kapitole 6.2. Vzniknou shluky, které mají dolní a horní aproximaci. Neurony v horních aproximacích jsou označeny jako za neurčité. Neurony v dolních aproximací jsou označeny za útok nebo běžný provoz podle toho, který typ neuronu převažuje. Při vlastním testu se prvky validační množiny Intruders posílají na vstup sítě. Porovnání, jestli se jedná o správnou klasifikaci vstupu se provádí podle příslušnosti výherního neuronu ve shluku či shlucích. Pokud se výherní neuron nachází v

dolní aproximaci shluku, je vstup porovnán s hodnotou celé dolní aproximace. Řekněme, že na vstup sítě je poslán prvek reprezentující útok. Vítězný neuron se nachází v dolní aproximaci shluku, kde převažují neurony reprezentující útok, tak klasifikace je úspěšná. Pokud vítězný neuron se nachází v horních aproximacích, tak klasifikace je neúspěšná. Měřené hodnoty byly zaznamenány v tabulkách 21,22,23 a 24 , které jsou pro svou velikost umístěny v příloze. Měřené hodnoty i s jejich zkratkami použité v tabulkách jsou uvedeny v tabulce 20. Pro velkou velikost byly tabulky s naměřenými hodnotami a zkratkami přesunuty do přílohy.

7.4.2 Hodnocení

Nejlepší hodnoty klasifikace množiny Intruders bylo naměřeno u sítě o 100 neuronech s threshold 0,1, s W_{lower} 0,65 a W_{upper} 0,35 o 6 shlucích. Algoritmus úspěšně klasifikoval 76,33% prvků množiny což je 5259 z celkového počtu 6890. Druhý nejlepší výsledek byl naměřen u sítě o 25 neuronech s nastavením thresholdu 0,1, s W_{lower} 0,9 a W_{upper} 0,1, který množinu rozdělil do 3 shluků a dosáhl tak úspěšnosti 75,08%.

8 Závěr

V této práci jsme měli za úkol prostudovat neuronové samoorganizující sítě se shlukováním využívající rough set a provést experimenty s jejich hodnocení. Jelikož jde o kombinaci několika algoritmů, bylo nutné je nejprve prostudovat. Práci jsme rozdělili do několika částí tak, aby na sebe navazovaly. Práce začíná postupným úvodem do problematiky neuronových sítí. Formou různých příkladů jsme se snažili co nejvíce tuto problematiku přiblížit. Objasnili jsme zjevnou spojitost neuronových sítí s neurony, které se nacházejí v mozku. Dále jsme uvedli základní rozdělení a příklady použití v praxi. Více jsme popsali problematiku samo organizujících sítí, protože byla základním stavebním kamenem této práce.

Dále jsme popsali shlukovací algoritmus K-means a sním spojené vzorce, pseudo kód a implementace v jazyce C# použitá v experimentech. Kde jsme se snažili názorně vysvětlit princip shlukování tohoto algoritmu. V kapitole Rough set jsme představili tuto teorii, uvedli její význam a problémy, které řeší. Po vysvětlení všech algoritmů jsme uvedli jak spolu jednotlivé části fungují a spolupracují.

Nejdůležitější část této práce je v provedených experimentech. Experiment s RGB barvami byl zamýšlen jako hlavní část této práce ale po konzultaci s vedoucím práce jsme množinu dat změnili na množinu, která je vhodnější pro shlukování a řeší reálný problém. Nicméně projekt byl v takové fázi vývoje, že jsme se rozhodli ho v této práci uvést pro jeho jednoduchou a přehlednou vizualizaci a ukázání vlivu jednotlivých parametrů na výsledek. Pro posouzení utvořených shluků jsme chtěli použít validační indexy Fukuyama-Sugeno a Xie-Beni. Ukázalo se však, že index Xie-Beni není jednoduchý na nastavení a museli bychom navíc ještě ověřovat jeho funkci, tedy pro další experimenty bylo rozhodnuto tyto indexy vypustit. Nicméně jsme je implementovali a použili v experiment s RGB barvami, kde postačilo základní nastavení. Před vlastním experimentem jsme blíže popsali kolekci dat, které jsme použili. Jedná se o kolekci, která charakterizuje síťový provoz na síťovém prvku, kde se podnikaly síťové útoky. Na základě těchto dat jsme provedli experimenty. Úkolem bylo co nejpřesněji rozlišit útok od běžného provozu. Měřili jsme úspěšnost tohoto rozlišení, počet správně určených útoků nebo běžného provozu a výsledky jsou v uvedených tabulkách.

Druhý experiment je měření úspěšnosti klasifikace množiny Intruders pomocí neuronových samoorganizačních sítí. Vytvořili jsme sítě s různými parametry, které jsme následně natrénovali testovací množinou Intruders a ohodnotili množinou validační ze stejné kolekce. Z naměřených hodnot lze vidět, že úspěšnost je relativně dobrá a pohybuje se ,až na pár případů, okolo 80%. Zřejmě by bylo možné dosáhnout lepších výsledků, kdyby jsme použili jiné parametry sítí či jinou topologii(například hexadecimální).

Třetí experiment je měření klasifikace množiny Intruders. Nyní jsme ale použili neuronové sítě a algoritmu K-means, který shlukoval neurony ve výstupní vrstvě sítí. Naměřené hodnoty se pohybují okolo 70%. Jen u některých kombinací parametrů dosahovala úspěšnost až k 85%. Přičina těchto né zcela přiznivých hodnot je, že algoritmus K-means inicializuje své počáteční centra náhodně. Návíc závisí na kvalitě natrénované sítě. Tudíž je možné dostat výsledky ve velkém rozsahu hodnot.

V posledním experimentu jsme měřili úspěšnost klasifikace množiny Intruders pomocí neuronových sítí a algoritmu K-means se zakomponovanými rough sety. Toto měření a implementace byla nejnáročnější v celé práci. Museli jsme vyřešit spolupráci všech částí. Dalším problémem byla otázka ohodnocení výsledků shlukování. Po konzultaci s vedoucím práce jsme rozhodli, že klasifikaci budeme provádět pomocí neuronů, které jsou v dolních aproximacích shluků. Neurony v horních aproximacích jsme prohlásili za neidentifikovatelné díky jejich příslušnosti do více shluků. Ve výsledcích tedy tyto neurony nejsou brány v potaz. Vzhledem k množství sítí a nastavovacích proměnných jsme parametry testů vybráli intuitivně podle zkušeností nabytých touto prací. Z tabulek výsledků měření lze na první pohled vidět, že úspěšnost klasifikace je velice špatná a že se pohybuje okolo 50%. Toto je způsobeno náhodnou inicializací shluků, ohodnocením klasifikace a málým rozsahem testovacích parametrů. Jak jsme řekli, ohodnocení probíhalo jen u neuronů v dolních aproximacích. Nejvyšší úspěšnost byla i přesto 76,33%. Pokud by se hodnotily i neurony v horních aproximacích, úspěšnost by byla daleko větší.

Největší problém, na který nás celou práci provázel byla určitá náhodnost ve všech prvcích, která měla nemalý vliv na konečné výsledky. Například prvotní inicializace vah neuronové sítě by měly být náhodně malé hodnoty. Dále způsob výběru prvků z testovací množiny v učící fázi má také jistý vliv. U algoritmů K-means samotného nebo v kombinaci s rough sety se také počáteční centra inicializují náhodně. Většina nastavovacích parametrů jsme tedy museli zjistit experimentálně, protože zde není jasná série definovaných kroků. Pro omezení tohoto vlivu by bylo vhodné aplikovat nějaký algoritmus, který pomůže nebo přímo určí ideální počet center či jejich pozici (například fiedlerův algoritmus). Dále pro dosažení co nejlepších hodnot je potřeba provést více měření s více hodnotami parametrů. Jelikož trénování sítí je výpočetně náročné, za zvážení by také stálo udělat aplikaci využívající více procesorů.

Bc. Martin Klapec

9 Reference

[1] *Teuvo Kohonen [online]*, Dostupný z WWW: `http://users.ics.tkk.fi/teuvo/`, [cit. březen 2012].

[2] Doug Alexander, *Data Mining [online]*, Dostupný z WWW: `http://www.laits.utexas.edu/~norman/BUS.FOR/course.mat/Alex/`, [cit. březen 2012].

[3] *Sigma AK222 [online]*, Dostupný z WWW: `http://mycalcdb.free.fr/main.php?l=0&id=4334`, [cit. únor 2012].

[4] *TOP500 supercomputer sites [online]*, Dostupný z WWW: `http://www.top500.org/`, [cit. duben 2012].

[5] *Neuronové sítě [online]*, Dostupný z WWW: `http://cgg.mff.cuni.cz/~pepca/prg022/mucha/`, [cit. březen 2012].

[6] Weisz O., *Základní logické funkce - AND [online]*, Dostupný z WWW: `http://www.edunet.souepl.cz/~weisz/dokuwiki/doku.php?id=logo:uvod:logicke_funkce`, [cit. březen 2012].

[7] Heaton, J., *Introduction to Neural Network for C#, 2nd Edition,* Heaton Research, Incorporated 2008, ISBN 978-1-6043-9009-4

[8] Gan,Guojun,Chaogun Ma, and Jiahong Wu *Data Clustering: Theory,Algorithms and Applications* ASA-SIAM Series on Statistics and Applied Probability, SIAM, Philadelphia, ASA, Alexandria, VA, 2007, ISBN 0898716233

[9] Wicher Martin *Shlukování pomocí COBWEB*, diplomová práce, 2010.

[10] Koukolík František *Lidský mozek. Funkcní systémy. Norma a poruchy* Galén, Práh, Praha 2000, 359 s, ISBN:978-80-7262-771-4

[11] Greco, Salvatore. *Rough Sets and Current Trends in Computing 5th International Conference*, RSCTC 2006, Kobe, Japan, November 6-8, 2006 : Proceedings. Berlin [etc.]: SpringerLink [host], 2006

[12] Mohebi, E.; Sap, M.N.M., *Hybrid Self Organizing Map for Overlapping Clusters*,Computer Modelling and Simulation, 2009. UKSIM '09. 11th International Conference on , vol., no., pp.53-58, 25-27 March 2009,`http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4809737&isnumber=4809715`

[13] Pawan Lingras, Chad West, *Interval Set Clustering of Web Users with Rough K-Means*, J. Intell. Inf. Syst. 23, 1 (July 2004), 5-16., 2004.

[14] Faber Vance, *Clustering and the Continuous k-Means Algorithm*, Los Alamos Science, 22, 138-144, 1994. Dopsat, 2003.

[15] Novotný Jiří, *Stroje a množiny Zdzdis lawa Pawlaka [online]*,Dostupný z WWW:http://math.fce.vutbr.cz/~pribyl/workshop_2006/prispevky/Novotny.pdf, 2012.

[16] Federico Marini *Artificial neural networks in foodstuff analyses: Trends and perspectives A review*Dipartimento di Chimica – Sapienza Universitŕ di Roma, P.le Aldo Moro 5, I-00185 Rome, Italy http://www.sciencedirect.com/science/article/pii/S0003267009000191, 2009.

[17] Micallef Timothy,Rossiter Jonathan *Human Inspired Image Classification [online]*, Dostupný z WWW: http://seis.bris.ac.uk/~tm7976/human.htm, [cit. duben 2012].

[18] *Lidské tělo [online]*, Dostupný z WWW: http://www.latinsky.estranky.cz/, [cit. duben 2012].

[19] Viscovery *Self-organizing maps [online]*, Dostupný z WWW: http://www.viscovery.net/self-organizing-maps, [cit. duben 2012].

[20] Math Works *K-means clustering [online]*, Dostupný z WWW: http://www.mathworks.com/help/toolbox/stats/kmeans.html, [cit. duben 2012].

[21] Subrajeet Mohapatra, Dipti Patra, Kundan KumarISRN *Unsupervised Leukocyte Image Segmentation Using Rough Fuzzy Clustering*,Artificial Intelligence http://www.isrn.com/journals/ai/2012/923946.fig.005.jpg, 2012.

A Tabulky

K	Počet shluků
GA	Počet správě klasifikovaných útoků
GN	Počet správě klasifikovaného běžného provozu
BA	Počet špatně klasifikovaných útoků
BN	Počet špatně klasifikovaného běžného provozu
TG	Celkový počet správně klasifikovaných vstupů
TG[%]	Celkový počet správně klasifikovaných vstupů v %
TB	Celkový počet špatně klasifikovaných vstupů
TB[%]	Celkový počet špatně klasifikovaných vstupů v %

Tabulka 15: Měřené hodnoty a jejich zkratky

Parametr učení	Epochy	K	GA	GN	BA	BN	TG	TG[%]	TB	TB[%]
0.2	200	3	3163	1257	1039	1431	4420	64,15	2470	35,85
		6	3209	1248	993	1440	4457	64,69	2433	35,31
		9	3259	1248	943	1440	4507	65,41	2383	34,59
		12	3259	1249	943	1439	4508	65,43	2382	34,57
0.2	500	3	4202	0	0	2688	4202	60,99	2688	39,01
		6	4028	762	174	1926	4790	69,52	2100	30,49
		9	2904	2099	1298	589	5003	72,61	1887	27,39
		12	3058	2366	1144	322	5424	**78,72**	1466	21,28
0.2	1000	3	3668	968	534	1720	4636	67,29	2254	32,71
		6	4201	776	1	1912	4977	72,24	1913	27,76
		9	4201	776	1	1912	4977	72,24	1913	27,76
		12	4201	779	1	1909	4980	72,28	1910	27,72
0.5	200	3	4202	0	0	2688	4202	60,99	2688	39,01
		6	4201	379	1	2309	4580	66,47	2310	33,53
		9	4201	380	1	2308	4581	66,49	2309	33,51
		12	4201	379	1	2309	4580	66,47	2310	33,53
0.5	500	3	4138	259	64	2429	4397	63,82	2493	36,18
		6	3004	2031	1198	657	5035	73,08	1855	26,92
		9	3004	2053	1198	635	5057	73,40	1833	26,60
		12	4169	1150	33	1538	5319	77,20	1571	22,80
0.5	1000	3	4119	312	83	2376	4431	64,31	2459	35,69
		6	4134	341	68	2347	4475	64,95	2415	35,05
		9	4123	949	79	1739	5072	73,61	1818	26,39
		12	4088	1266	114	1422	5354	**77,71**	1536	22,29

Tabulka 16: Naměřené hodnoty klasifikace algoritmu K-means a neuronové sítě o 25 neuronech

Parametr učení	Epochy	K	GA	GN	BA	BN	TG	TG[%]	TB	TB[%]
0.2	200	3	2589	1961	1613	727	4550	66,04	2340	33,96
		6	4202	5	0	2683	4207	61,06	2683	38,94
		9	4165	725	37	1963	4890	70,97	2000	29,03
		12	4202	5	0	2683	4207	61,06	2683	38,94
0.2	500	3	3056	1342	1146	1346	4398	63,83	2492	36,17
		6	3104	1341	1098	1347	4445	64,51	2445	35,49
		9	2860	1703	1342	985	4563	66,23	2327	33,77
		12	2838	2295	1364	393	5133	**74,50**	1757	25,50
0.2	1000	3	4202	0	0	2688	4202	60,99	2688	39,01
		6	4202	4	0	2684	4206	61,04	2684	38,96
		9	4202	0	0	2688	4202	60,99	2688	39,01
		12	4202	414	0	2274	4616	67,00	2274	33,00
0.5	200	3	3908	501	294	2187	4409	64,00	2481	36,00
		6	3699	1002	503	1686	4701	68,23	2189	31,77
		9	3903	1003	299	1685	4906	71,20	1984	28,80
		12	3903	1004	299	1684	4907	71,22	1983	28,78
0.5	500	3	4173	678	29	2010	4851	70,41	2039	29,60
		6	3766	977	436	1711	4743	68,84	2147	31,16
		9	2934	2358	1268	330	5292	76,81	1598	23,19
		12	2989	1423	1213	1265	4412	64,03	2478	35,97
0.5	1000	3	3806	746	396	1942	4552	66,07	2338	33,93
		6	2867	2231	1335	457	5098	73,99	1792	26,00
		9	2526	2595	1676	93	5121	**74,33**	1769	25,67
		12	2849	2272	1353	416	5121	**74,33**	1769	25,67

Tabulka 17: Naměřené hodnoty klasifikace algoritmu K-means a neuronové sítě o 49 neuronech

Parametr učení	Epochy	K	GA	GN	BA	BN	TG	TG[%]	TB	TB[%]
0.2	200	5	4201	187	1	2501	4388	63,67	2502	36,31
		10	3808	965	394	1723	4773	69,27	2117	30,73
		15	3714	1513	488	1175	5227	75,86	1663	24,14
		20	3430	1275	772	1413	4705	68,29	2185	31,71
		25	2949	2553	1253	135	5502	79,84	1388	20,16
		30	3010	2077	1192	611	5087	73,83	1803	26,17
0.2	500	5	2604	2030	1598	658	4634	67,26	2256	32,74
		10	4199	16	3	2672	4215	61,18	2675	38,82
		15	4201	348	1	2340	4549	66,02	2341	33,98
		20	2604	2032	1598	656	4636	67,29	2254	32,71
		25	4159	716	43	1972	4875	70,75	2015	29,25
		30	3802	1762	400	926	5564	80,75	1326	19,25
0.2	100	5	3576	2018	626	670	5594	81,19	1296	18,81
		10	3431	2383	771	305	5814	**84,38**	1076	15,62
		15	3576	2019	626	669	5595	81,20	1295	18,80
		20	3430	2395	772	293	5825	**84,54**	1065	15,46
		25	3437	2381	765	307	5818	**84,44**	1072	15,56
		30	3437	2381	765	307	5818	**84,44**	1072	15,56
0.5	200	5	4202	0	0	2688	4202	60,99	2688	39,01
		10	4193	14	9	2674	4207	61,06	2683	38,94
		15	2714	2320	1488	368	5034	73,06	1856	26,94
		20	2979	1608	1223	1080	4587	66,57	2303	33,43
		25	2782	1993	1420	695	4775	69,30	2115	30,70
		30	2782	1993	1420	695	4775	69,307	2115	30,70
0.2	500	5	2582	1993	1620	695	4575	66,40	2315	33,60
		10	2548	2006	1654	682	4554	66,10	2336	33,90
		15	4147	451	55	2237	4598	66,73	2292	33,27
		20	2582	1995	1620	693	4577	66,43	2313	33,57
		25	2582	2009	1620	679	4591	66,63	2299	33,37
		30	4147	456	55	2232	4603	66,81	2287	33,19
0.2	1000	5	1929	2671	2273	17	4600	66,76	2290	33,24
		10	3365	1461	837	1227	4826	70,04	2064	29,96
		15	3697	1794	505	894	5491	79,70	1399	20,30
		20	3365	1455	837	1233	4820	69,96	2070	30,04
		25	3243	2652	959	36	5895	**85,56**	995	14,44
		30	3365	1461	837	1227	4826	70,04	2064	29,96

Tabulka 18: Naměřené hodnoty klasifikace algoritmu K-means a neuronové sítě o 100 neuronech

Parametr učení	Epochy	K	GA	GN	BA	BN	TG	TG[%]	TB	TB[%]
0.2	200	5	4197	46	5	2642	4243	61,58	2647	38,42
		10	3268	1596	934	1092	4864	70,60	2026	29,40
		15	4197	198	5	2490	4395	63,79	2495	36,21
		20	4197	399	5	2289	4596	66,71	2294	33,29
		25	3269	1952	933	736	5221	75,78	1669	24,22
		30	4197	406	5	2282	4603	66,81	2287	33,19
0.2	500	5	4202	0	0	2688	4202	60,99	2688	39,01
		10	4201	189	1	2499	4390	63,72	2500	36,28
		15	3728	2144	474	544	5872	**85,22**	1018	14,78
		20	4126	1669	76	1019	5795	**84,11**	1095	15,89
		25	3737	2121	465	567	5858	**85,02**	1032	14,99
		30	4119	1709	83	979	5828	**84,59**	1062	15,41
0.2	1000	5	3753	2034	449	654	5787	83,99	1103	16,01
		10	3753	2034	449	654	5787	83,99	1103	16,01
		15	3664	2086	538	602	5750	83,45	1140	16,55
		20	3794	1954	408	734	5748	83,43	1142	16,57
		25	3754	2036	448	652	5790	**84,03**	1100	15,97
		30	3754	2036	448	652	5790	**84,03**	1100	15,97
0.5	200	5	3385	1184	817	1504	4569	66,31	2321	33,69
		10	3395	1460	807	1228	4855	70,46	2035	29,54
		15	3115	2085	1087	603	5200	75,47	1690	24,53
		20	3439	1743	763	945	5182	75,21	1708	24,79
		25	3395	1856	807	832	5251	76,21	1639	23,79
		30	3395	1439	807	1249	4834	70,16	2056	29,84
0.5	500	5	3454	1380	748	1308	4834	70,16	2056	29,84
		10	3488	1724	714	964	5212	75,65	1678	24,35
		15	3488	1383	714	1305	4871	70,70	2019	29,30
		20	3488	1724	714	964	5212	75,65	1678	24,35
		25	3453	1725	749	963	5178	75,15	1712	24,85
		30	3488	1755	714	933	5243	76,10	1647	23,90
0.5	1000	5	3640	628	562	2060	4268	61,94	2622	38,06
		10	4163	691	39	1997	4854	70,45	2036	29,55
		15	3562	1151	640	1537	4713	68,40	2177	31,60
		20	4117	700	85	1988	4817	69,91	2073	30,09
		25	4159	685	43	2003	4844	70,30	2046	29,70
		30	4160	664	42	2024	4824	70,01	2066	29,99

Tabulka 19: Naměřené hodnoty klasifikace algoritmu K-means a neuronové sítě o 196 neuronech

K	Počet shluků
GA	Počet správě klasifikovaných útoků
GN	Počet správě klasifikovaného běžného provozu
UA	Počet neidentifikovaných útoků
UN	Počet neidentifikovaného běžného provozu
TG	Celkový počet správně klasifikovaných vstupů
TG[%]	Celkový počet správně klasifikovaných vstupů v %
TU	Celkový počet neidentifikovaných vstupů
TU[%]	Celkový počet neidentifikovaných vstupů v %

Tabulka 20: Měřené hodnoty a jejich zkratky

Threshold	W_{lower} / W_{upper}	K	GA	GN	UA	UN	TG	TG [%]	TU	TU [%]
0.1	0,65 / 0,35	2	2634	1524	1568	1164	4158	60,35	2732	39,65
		3	2091	1524	2111	1164	3615	52,47	3275	47,53
		4	2634	1526	1568	1162	4160	60,38	2730	39,62
0.1	0,9 / 0,1	2	2634	1524	1568	1164	4158	60,35	2732	39,65
		3	2634	2539	1568	149	5173	**75,08**	1717	24,92
		4	2634	1524	1568	1164	4158	60,35	2732	39,65
0.2	0,65 / 0,35	2	2634	1511	1568	1177	4145	60,16	2745	39,84
		3	2634	1511	1568	1177	4145	60,16	2745	39,84
		4	2634	1524	1568	1164	4158	60,35	2732	39,65
0.2	0,9 / 0,1	2	2634	1524	1568	1164	4158	60,35	2732	39,65
		3	2634	2524	1568	164	5158	**74,86**	1732	25,14
		4	2634	1511	1568	1177	4145	60,16	2745	39,84

Tabulka 21: Naměřené hodnoty klasifikace algoritmu Rough Set a neuronové sítě o 25 neuronech s parametrem učení 0,2 a 1000 epochách

Threshold	W_{lower} / W_{upper}	K	GA	GN	UA	UN	TG	TG [%]	TU	TU [%]
0.1	0,65 / 0,35	2	3987	0	215	2688	3987	57,87	2903	42,13
		3	3987	0	215	2688	3987	57,87	2903	42,13
		4	3987	0	215	2688	3987	57,87	2903	42,13
0.1	0,9 / 0,1	2	3987	0	215	2688	3987	57,87	2903	42,13
		3	3921	943	281	1745	4864	**70,6**	2026	29,4
		4	3987	0	215	2688	3987	57,87	2903	42,13
0.2	0,65 / 0,35	2	3921	943	281	1745	4864	70,6	2026	29,4
		3	3987	0	215	2688	3987	57,87	2903	42,13
		4	3987	943	215	1745	4930	71,55	1960	28,45
0.2	0,9 / 0,1	2	3987	943	215	1745	4930	**71,55**	1960	28,45
		3	3921	943	281	1745	4864	70,6	2026	29,4
		4	3920	0	282	2688	3920	56,89	2970	43,11

Tabulka 22: Naměřené hodnoty klasifikace algoritmu Rough Set a neuronové sítě o 49 neuronech s parametrem učení 0,5 a 1000 epochách

Threshold	W_{lower} / W_{upper}	K	GA	GN	UA	UN	TG	TG [%]	TU	TU [%]
0.1	0,65 / 0,35	6	3135	2058	1067	630	5193	**75,37**	1697	24,63
		8	2884	2197	1318	491	5081	73,74	1809	26,26
		12	3156	1305	1046	1383	4461	64,75	2429	35,25
0.1	0,9 / 0,1	6	3134	2125	1068	563	5259	**76,33**	1631	23,67
		8	3176	1314	1026	1374	4490	65,17	2400	34,83
		12	3157	1315	1045	1373	4472	64,91	2418	35,09
0.2	0,65 / 0,35	6	2733	2197	1469	491	4930	71,55	1960	28,45
		8	1615	2191	2587	497	3806	55,24	3084	44,76
		12	1987	1314	2215	1374	3301	47,91	3589	52,09
0.2	0,9 / 0,1	6	3157	1309	1045	1379	4466	64,82	2424	35,18
		8	3170	1304	1032	1384	4474	64,93	2416	35,07
		12	1962	1310	2240	1378	3272	47,49	3618	52,51

Tabulka 23: Naměřené hodnoty klasifikace algoritmu Rough Set a neuronové sítě o 100 neuronech s parametrem učení 0,5 a 500 epochách

Threshold	W_{lower} / W_{upper}	K	GA	GN	UA	UN	TG	TG [%]	TU	TU [%]
0.1	0,65 / 0,35	6	2592	684	1610	2004	3276	47,55	3614	52,45
		8	3877	501	325	2187	4378	**63,54**	2512	36,46
		12	2903	810	1299	1878	3713	53,89	3177	46,11
0.1	0,9 / 0,1	6	2917	268	1285	2420	3185	46,23	3705	53,77
		8	2592	1057	1610	1631	3649	52,96	3241	47,04
		12	2906	795	1296	1893	3701	53,72	3189	46,28
0.2	0,65 / 0,35	6	3878	0	324	2688	3878	56,28	3012	43,72
		8	2969	264	1233	2424	3233	46,92	3657	53,08
		12	2915	1069	1287	1619	3984	57,82	2906	42,18
0.2	0,9 / 0,1	6	2375	1806	1827	882	4181	60,68	2709	39,32
		8	2592	1055	1610	1633	3647	52,93	3243	47,07
		12	2917	266	1285	2422	3183	46,2	3707	53,8

Tabulka 24: Naměřené hodnoty klasifikace algoritmu Rough Set a neuronové sítě o 196 neuronech s parametrem učení 0,2 a 500 epochách

B Obsah CD

- text práce
- zdrojové kódy s implementací experimentů
- kolekce intruders